UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

TREBALL DE FI DE GRAU

TFG TITLE: 3D CFD analysis of generation of bubbles in a double T-junction mini-channel

DEGREE: Grau en Enginyeria d'Aeronavegació

AUTHORS: Amina Bakkali Abderrahaman

ADVISORS: Santiago Arias Calderón
Ming-Jyh Chern

DATE: July 18, 2018

Título: Análisis 3D CFD de la generación de burbujas en un mini-canal doble unión T

Autores: Amina Bakkali Abderrahaman

Directores: Santiago Arias Calderón
Ming-Jyh Chern

Fecha: 18 de julio de 2018

Resumen

El tema de los flujos de dos o de varias fases se ha vuelto cada vez más poderoso en una amplia variedad de sistemas de ingeniería. El objetivo de nuestro proyecto es estudiar la generación de burbujas de la combinación agua-aire en un capilar de unión T, que se comparará con una doble unión T, en condiciones de microgravedad debido a un comportamiento muy diferente al observado en presencia de fuerzas gravitacionales.

Para alcanzar nuestro objetivo, hemos preparado todos los parámetros y el marco (geometría, malla, condiciones de contorno, etc.) para realizar las simulaciones con el programa de código abierto OpenFOAM. Además, hemos llevado a cabo diferentes pruebas de convergencia calculando los errores y las desviaciones estándar de la frecuencia, velocidad, longitud y volumen de las burbujas. En adición, hemos trabajado en numerosas pruebas para mejorar la calidad de generación de burbujas, probando distintas condiciones de contorno, como el ángulo de contacto, la *wettability*, las capas límite en la malla, etc. Terminamos usando la condición de *wettability* para el capítulo de resultados, ya que resolvió el problema más urgente, que es la unión de burbujas a las paredes. También hemos utilizado una capa límite en la malla porque disminuyó las fluctuaciones de los parámetros de las burbujas.

Como conclusión importante, la combinación de las velocidades superficiales del aire y el agua tiene un gran efecto en los resultados obtenidos y que algunas condiciones de contorno pueden ser óptimas para algunas de las combinaciones utilizadas, pero lo contrario para algunas otras. Por ejemplo, la generación de burbujas se vuelve irregular con altas fluctuaciones si la velocidad superficial del gas es más alta que la del líquido, pero se vuelve más estable si las velocidades son similares o si el líquido es más rápido. En cuanto a los resultados finales, se dedujo que la doble unión T es una geometría más óptima ya que, para velocidades superficiales líquidas más altas, la generación de burbujas era más estable debido a su simetría de dos entradas verticales de agua.

Por último, La condición del ángulo de contacto, todavía necesita más investigación debido a la vaguedad que gira entorno a sus numerosas pruebas. Sin olvidar que, dado que la doble unión en T es una geometría recientemente introducida en este análisis, también necesitará más evaluaciones para ser totalmente aceptada como una mejor opción.

Title : 3D CFD analysis of generation of bubbles in a double T-junction mini-channel

Authors: Amina Bakkali Abderrahaman

Advisors: Santiago Arias Calderón
Ming-Jyh Chern

Date: July 18, 2018

Overview

The subject of two or multi-phase flows has become increasingly powerful and important in a wide variety of engineering systems. The aim of our project is to study the bubble generation of the water-air combination in a T-junction capillary, which will be compared to a double T-junction, under microgravity conditions due to the very different behavior when compared to the one observed in the presence of gravitational forces.

To reach our goal, we have prepared all the parameters and framework (geometry, mesh, boundary conditions, etc.) to perform the simulations with the open-source program OpenFOAM. We have also carried out different convergence tests by computing the errors and the standard deviations of the bubble generation frequency, velocity, length, and volume. In addition, we have worked on numerous tests to enhance the bubble generation quality, by trying different boundary conditions such as the contact angle, the wettability, boundary layers in the mesh, etc. For the chapter of results, we ended up using the wettability condition, since it solved the most urgent issue, which is the bubble attachment to the walls. We have also used a boundary layer in the mesh because it decreased the fluctuations of the bubble parameters.

One of the most important conclusions is that the combination of the air and water superficial velocities have a great effect on the obtained results and that some boundary conditions can be optimal for some of the used combinations but the opposite for some others. For example, the bubble generation grows irregular with high fluctuations if the gas superficial velocity is higher than the liquid's, but becomes more stable if the velocities are similar or if the liquid is faster. As for the final results, it was deduced that the double T-junction is a more optimal geometry since, for higher liquid superficial velocities, the bubble generation was more stable due to its two vertical water entries symmetry.

Lastly, the contact angle condition still needs more research due to vagueness revolving around its numerous tests. Not to forget that, since the double T-junction is a newly introduced geometry in this CFD two-phase flow analysis, it will also need more evaluations to fully be accepted as a better option.

To my family who encouraged me during my stay far from home, to my professors Arias and Ming-Jyh who gladly guided me and supported my work, to my laboratory colleagues who always tried their best to solve my doubts.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Dimensionless numbers

B_o	Bond number	
C_a	Capillary number	
C_o	Courant number	
R_e	Reynolds number	
W_e	Weber number	

Greek symbols

α	Air fraction	
ϵ	Error	$\%$
σ	Surface tension at the air-water interface	N/m
θ	Contact angle	degrees ^{o}

Rest of symbols

Δt	Time step	s
Δx	Size of the mesh grid	m
μ_G	Dynamic viscosity of air	Pa
μ_L	Dynamic viscosity of water	Pa
ϕ_C	Diameter of pipe	m
ρ_G	Density of air	Kg/m3
ρ_L	Density of water	Kg/m^3
σ_{f_B}	Standard deviation of the bubble frequency	Hz
σ_{L_B}	Standard deviation of the bubble length	m
σ_{U_B}	Standard deviation of the bubble velocity	m/s
σ_{V_B}	Standard deviation of the bubble volume	m^3
f_B	Bubble generation frequency	Hz
f_{sat}	Saturation frequency	Hz
g	Gravity	m/s^2
L	Length of pipe	m
L_B	Bubble length	m
T_B	Bubble generation period	s^{-1}
t_B	Bubble duration	s
T_C	Convective time unit	time units
U_B	Bubble velocity	m/s

U_M	Mixture of air-water superficial velocities	m/s
U_{SG}	Superficial velocity of air	m/s
U_{SL}	Superficial velocity of water	m/s
V_B	Bubble volume	m^3

ACKNOWLEDGMENTS

First of all, I would like to thank my friend and classmate Blanca Dalfó Ferrer who faced with me all the struggles related to our researches, which share the same subject, and also all the good and bad things of our experience together in Taiwan. I would also like to thank a lot our professors Dr. Ming-Jyh Chern of the National Taiwan University of Science and Technology, and Dr. Santiago Arias Calderón of the Polytechnic University of Catalunya for their constant encouragement and guidance during all the phases of my thesis. All thought Dr. Arias have supervised us from Spain, he has always kept himself aware of everything, in order to avoid my deviation from the important aspects of the project. He has also shown great flexibility with the Skype meeting every time we faced some difficulties. As for Dr. Ming, he has made sure that we would work in good conditions by introducing me to our laboratory colleagues who have continuously tried their best to solve my doubts, especially Amber Huang and Jason Kao. He has also regularly kept an eye on us by having the group and individual meetings every week to ensure the correct development of our study.

Secondly, I would like to express my gratitude to the PhD student Bahnu Prakash from the Polytechnic University of Catalunya who has helped us not to only learn about the open-source program OpenFOAM that we, at first, knew nothing about how to use it for our numerical simulations, but also all the important principles related to the CFD simulations. He kindly helped us to take our first steps into the world of the computational fluid dynamics.

Finally, I am truly grateful to my parents because they have always kept me connected to them as if I was still at home. Also, I should not forget the rest of my family members in Morocco and Spain who have always cheered me up and kept in touch without getting used to my absence. Not to forget that they would, from time to time, ask me to explain my study and progress so that they could make silly jokes about it, when I was facing some complications, in order to make it less weightful for me.

INTRODUCTION

Thanks to the numerous studies dedicated to the world of biphasic fluids, very promising results are being obtained. In the space sector, the need for new improvements in the different systems required for space missions has been increasing in the recent decades. One of the most important factors in the space industry is the weight, and biphasic fluids stand out with excellent results in weight reduction, which is way far from being achieved with monophasic fluids. Moreover, two-phase flows have great importance in the thermal management systems in terrestrial and space applications. For the normal gravity cases, the gas-liquid flows have been studied by the petroleum and nuclear industries. The petroleum industry has focused most of their efforts on flow through long pipelines with the intent of transferring a mixture of crude oil and then perform the separation of the components. The nuclear industry has been concerned with the system stability and safety in order to prevent, for example, the dry-out of the nuclear reactors due to either a heat instability or the loss of coolant. As for a reduced-gravity environment, for the most part, several systems in Earth orbit rely on photovoltaic single-phase systems, however there have been studies indicating that for a power system requiring more than 25 kW, a solar dynamic Rankine two-phase system is much more efficient in terms of the launch mass of the system and the propellant required to overcome any drag encountered on the devices used to collect the solar energy [1]. One another important example is the long-term propellant storage. Not to forget that the life support systems for human exploration of space are one of the most powerful examples, such as the water recovery systems for long-duration missions, humidity and temperature control within the environments of a crewed vehicle for comfort and safety, special plants for food, consumption of carbon dioxide and oxygen production [2]. Consequently, there are many applications of the biphasic fluids, and many more that are still being tested. They are a key factor that will guide numerous projects and studies to powerful and efficient results in various fields.

The simulations of our study will be performed in a microgravity environment due to the major role that gravity plays in governing the behavior of the two-phase flows, which is because of the large density difference between the phases. The gravitational body forces on earth dominate the gas-liquid flows, however, when gravity is reduced to almost zero, the gravitational body force becomes minimal. For example, in the case of thermal management systems, since the two-phase flows and the heat transfer characteristics change in a buoyancy-free environment, extensive tests of the thermal performance of different space systems need to be done. One another reason to prove the need of more research related to low-gravity conditions is that the flow patterns have been reasonably well-defined for medium and large diameter pipes, however for very small diameters research has shown that the surface tension forces become important, and the patterns, in the absence of gravity, are not as well-defined as in the case of gravity.

We will dedicate our work to the combination of air and water, in micro-tubes (1-millimeter diameter), because they are common fluids that are less complex in comparison with others. Another reason is the existing experiments that have already used these same fluids, which our results will be compared to. We will study the interaction between the liquid and the gas in order to generate a train of bubbles. We have chosen a T-junction configuration previously studied by other students and researchers. We will

also introduce a double T-junction, which will be compared with the experimental and numerical results of the normal T-junction in order to add more results to the previous studies of Arias [4] [3] related to this subject. We can see the different representations of the perpendicular pipelines in the figures 1 and 2.

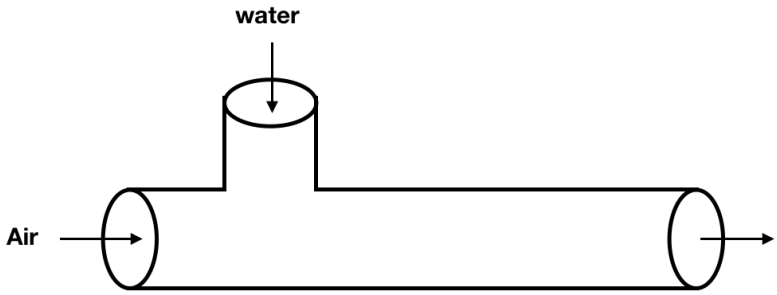

Figure 1: Representation of the air-water T-junction

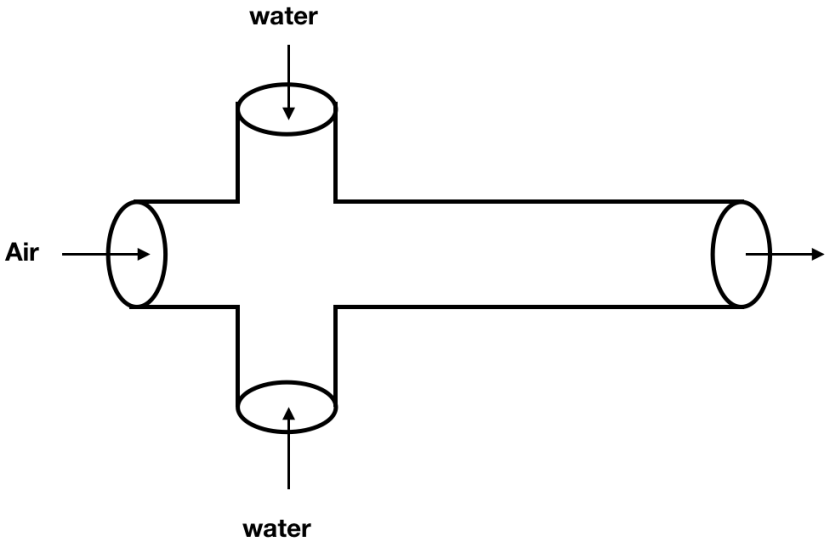

Figure 2: Representation of the double T-junction

As we can see in the figure 1, air enters the pipeline from the horizontal inlet and the water from the vertical one, to meet at a certain point, where the bubbles will be generated, primarily, because of the competition between the capillary and the liquid drag forces. This is mainly due to the surface tensions which are bigger than the buoyancy and inertial forces in a low-gravity environment.

We will focus on studying and analyzing the values of the bubble generation frequency f_B, the bubble velocity U_B, and finally the bubble length L_B. These values will be compared for the cases described previously in order to see the similarities and differences. In addition, we will use different combinations of velocities to see which combinations facilitate or difficult the bubble generation process. Moreover, different contact angles will be tested in order to find the most suitable value.

As for the report, we will start with the methodology chapter where an explanation of the CFD program and a brief description of the 3D geometry will be given. Secondly, we will explain the mesh configuration that plays an important role to obtain the desired results. The next step will be the description of the characteristics of the fluids as well as the different parameters affecting the study and simulations. In addition, we will proceed with the explanation of the boundary, initial and contact angle conditions, followed by the mathematical study of the project. For the validations chapter, we will discuss the convergence tests in order to choose the most suitable mesh, the different tests to enhance the bubble generation, and finally some tests for the double T-junction. The next chapter will describe and analyze the obtained results for different combinations of superficial velocities. Lastly, we will give a conclusion of the all the tests and results, as well as the future work to be done as a continuation of this project.

CHAPTER 1. METHODOLOGY

A successful CFD simulation depends on many factors such as the well-defined boundary conditions, differential partial equations, the mesh quality, etc. The geometry, as well as the mesh creation process, will both be explained in this chapter. They will be followed by the description of the fluids, the simulation parameters, and the mathematical study.

1.1. The mathematical study

In this subsection, we will see the main mathematical equations that describe our a multi-phase simulation project. We will start with the following equation of continuity since our fluids are incompressible. This section was based on the article [8] and the presentation [7]. We have also read a chapter bout the viscous flows of the book [9] to gain more knowledge on this field.

$$\nabla \cdot \mathbf{U} = 0 \tag{1.1}$$

Since it is a multi-phase flow, we use the Volume Of Fluid (VOF) method. In this type of modeling, the governing equations are solved for each separated phase (air and water), and the interface has to be tracked somehow. Together with the solution to these equations the reconstruction of the interface is made. Furthermore, some extra parameters are needed such as surface tension and the contact angle. At first, we have not considered any contact angle for the convergence tests since it is still under complex studies and investigations. However, for some combinations of the gas and liquid superficial velocities, a contact angle was needed in order to generate bubble results.

We will solve the following incompressible version of the Navier-Stokes Momentum equation for viscous flows at microgravity conditions:

$$\frac{\partial}{\partial t}(\rho\ \mathbf{U}) + \nabla \cdot (\rho\ \mathbf{UU}) = -\frac{\partial p}{\partial x} + \mu\ \triangle \mathbf{U} + \mathbf{F_s} + \mathbf{F_i} + \mathbf{F_m} \tag{1.2}$$

where F_i and F_m are respectively the inter-phase (drag and lift) and mass forces acting on the phases, which will be both neglected since we don't have turbulence nor gravity. F_s is the surface tension of the liquid, which is defined as:

$$F_s = \sigma\ K\ \triangle \alpha \tag{1.3}$$

where $\sigma = 0,072\ N/m$ is the tension coefficient, α is the average of all the air fractions of every grid in a cross-sectional area, and K is the mean curvature of the free surface, determined from the expressions and it is defined as follows:

$$K = -\nabla \cdot \frac{\nabla \alpha}{|\ \nabla \alpha\ |} \tag{1.4}$$

With the VOF method we will calculate a mean average density as follows:

$$\rho = \alpha\ \rho_G + (1-\alpha)\ \rho_L \tag{1.5}$$

If $\alpha = 1$ then in that cell we take the density of the gas, if $\alpha = 0$ then we take the density of water, and if $0 < \alpha < 1$ then the cell contains an interface of both phases. In addition, we transport this field with the following transport equation, and we advect it with an additional term for interface compression:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha\ \mathbf{U}) + \nabla \cdot (\alpha\ (1-\alpha)) \tag{1.6}$$

Finally, This model does not take into account the heat (and mass) transfer. The flow is isothermal so we will not have an energy equation.

Observation: These equations cannot be simply modified in **OpenFOAM**, instead, there are many cases for two-phase flows that this program has as tutorial folders which can be used for each specific study, with all the partial differential equation already defined.

1.2. CFD program

1.2.1. OpenFOAM

For this project, we have used the **OpenFoam** program as the main tool for our two-phase flow simulations for the reason that it is an open-source program, which makes it free and modifiable. It allows to fully understand the behind scenes of every studied case, it allows to define and control every condition such as boundaries, meshing, the mathematical differential equations, etc. One of the disadvantages is that it is not a friendly program since it does not have a visual interface, therefore, it is more difficult to understand and use in the beginning. The programmer's guide [5] is hopeful to have a deep introduction to OpenFOAM. The figure 1.1 shows a general schematic view of the OpenFOAM folders, which can be found in Appendix A in more detail. In addition, the OpenFOAM simulations follow the following main phases:

Pre-processing:

In this first step, we define all the necessary conditions and equations for the simulations. For example, we define the boundary and initial conditions of the pressure, velocity and air fraction in the "0" folder. As for the "Constant" folder, we define the characteristics of our fluids in the "transportPropert" file, the mesh in the "polyMesh" folder, etc. in the "system" folder we define, for example, the time step and monitors in the "controlDict" file, the initial volume of gas in the "setFieldsDict" file, etc.

Processing:

In this phase, we run the simulations with the InterFoam solver for 2 incompressible fluids. It tracks the interface and includes the option of mesh motion. One important

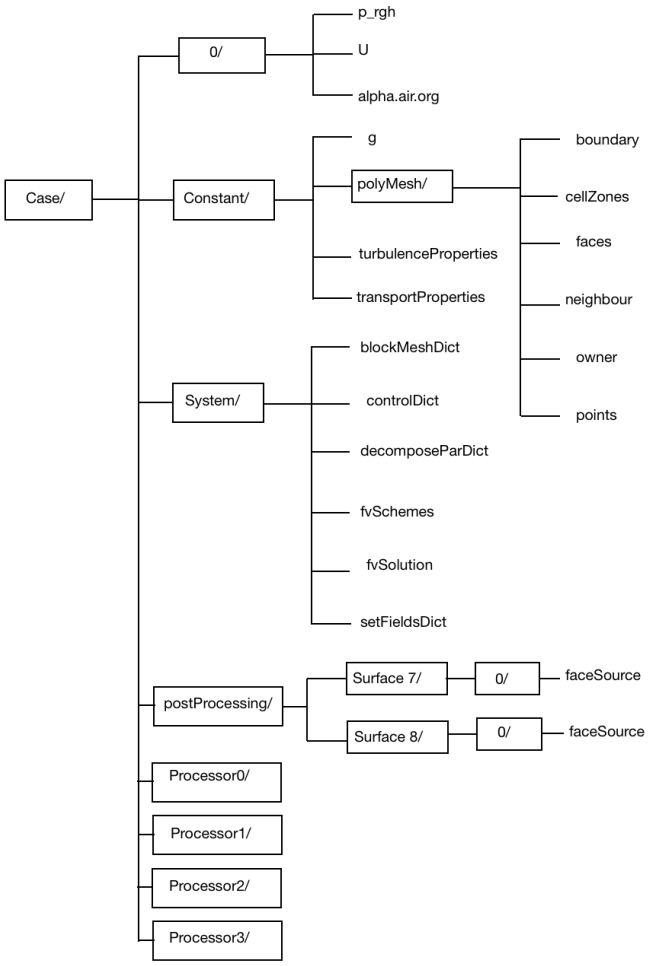

Figure 1.1: A diagram of the OpenFOAM folders

thing is that, when the simulations start, the OpenFOAM folders cannot be modified. Also, the simulations can be fast or slow depending on the chosen time step, the mesh, the gas and liquid superficial velocities, etc.

Post-processing:

In this last step, we take a "Post-processing" folder that appears inside the Case after starting the simulations and analyzes the data. The folder contains whatever parameter we ask for inside the "controlDict" file such as the air fraction α, the pressure, etc. We use **Paraview** to visualize not only the geometry and the mesh but also the different parameters such as the pressure and velocity, and also the animations of the bubble generation. We compute and plot the bubble parameters with **Matlab**.

1.2.2. OpenFOAM parallelism

The OpenFOAM performance can be lower than some other programs such as ANSYS. However, the simulations can be faster if we run the program in parallel on distributed processors. This method of parallel computing is known as domain decomposition, in which the geometry and associated fields are decomposed into pieces and allocated in

separated processors for the solution. The aim of this method is to break up the domain with minimal effort but in such a way to guarantee an efficient and fast solution.

Two short simulations were performed with a mesh of approximately 315.000 cells for 1, 2, 4 and then 8 cores. All through the increase in the number of cores makes the simulations to be faster, the communication between the machines uses Ethernet and this causes the increase of the simulation time between the cores. As a result, many tests will have to be done in order to check which number of cores is best to perform the parallel decomposed simulations. We have obtained the following results:

- For 1 core: The simulation ended after **1507 seconds** of execution time and **1501,62 seconds** of clock time.
- For 2 cores: The simulation ended after **874,38 seconds** of execution time and **880 seconds** of clock time.
- For 4 cores: The simulation ended after **565,57 seconds** of execution time and **580 seconds** of clock time.
- For 8 cores: The simulation ended after **851,56 seconds** of execution time and **1707 seconds** of clock time.

The conclusion is that **the decomposed simulation of 4 cores is the best option** since it is the fastest one.

1.3. Description of the 3D geometry

In this section, we describe the geometries, which are the combination of two perpendicular cylinders that result in a T-junction or a double T-junction. The diameter of these pipes is 1 mm and the total length is 10 mm. The **OnShape** program was used to create the solid geometry of the T-junction. This software returns the STereoLithography (STL) files that can be read by many other CAD programs. Besides, almost any type of modification can be done to the geometry in need without any difficulty, the dimensions for example. The geometry was emptied with the program **SALOME** by Blanca Dalfó in her Windows computer. One of the disadvantages of SALOME is the high-performance requirement from the host computer, in addition to its incompatibility with my Mac computer. Consequently, the double T-junction, which is my geometry was created with **Solidworks**, since allowed to directly create empty perpendicular pipes with solid intersections. To have a completely empty configuration, the information of these solid intersections was manually deleted from the STL text file generated with Solidworks. We can see the 3D geometries and dimensions in the figure 1.2.

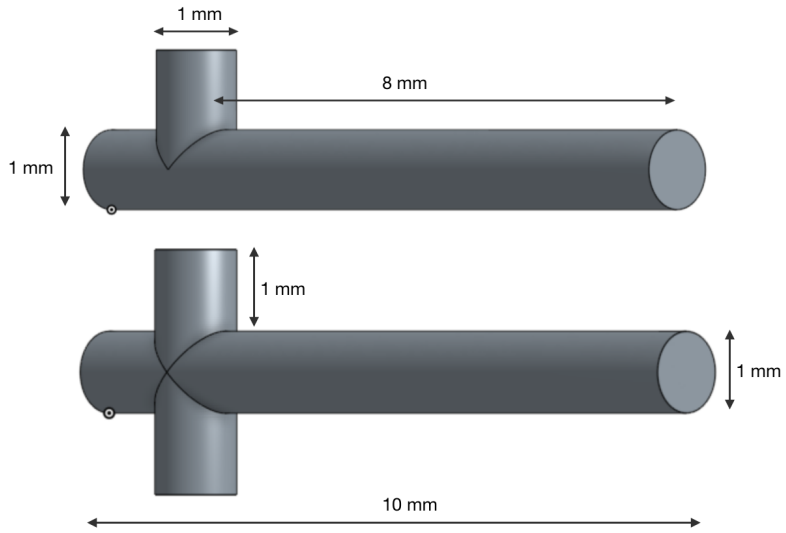

Figure 1.2: 3D lateral view of the T and double T junctions with the dimensions

1.4. The mesh

Many difficulties were faced in the progress of finding the best software to mesh since the geometry was a little bit complex. Salome, blockMesh, and snappyHexMesh were some of the tools we tried at first. On one hand, blockMesh does not support complicated geometries such as a T-junction, and snappyHexMesh was difficult to use. On the other hand, Salome is a good option for meshing but a bad one to create the geometry. Then we found SimScale to be the most suitable software. SimScale is a CAE software product based on cloud computing and uses open source codes. It allows Computational Fluid Dynamics, Finite Element Analysis, and Thermal simulations. However, our simulations could be completed because of the Courant number which was directly related to the size of the mesh grids and the time step.

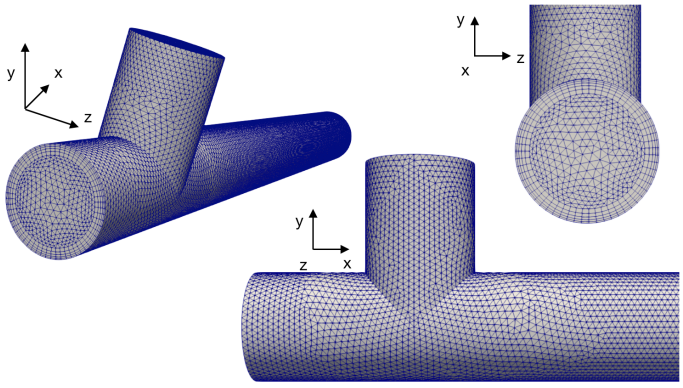

Figure 1.3: 3D representation of an example of the simScale mesh

To sum up, the errors were due to our mesh, hence we needed to find another program to create it. Finally, we have ended up using **ICEM**, which is a sub-part of ANSYS related to the phase of mesh creation. We have created structured meshes with **hexahedral** elements. The advantage is that the cells had controllable sizes, which could not cause the sudden increase of the Courant number that will be explained in the next chapter.

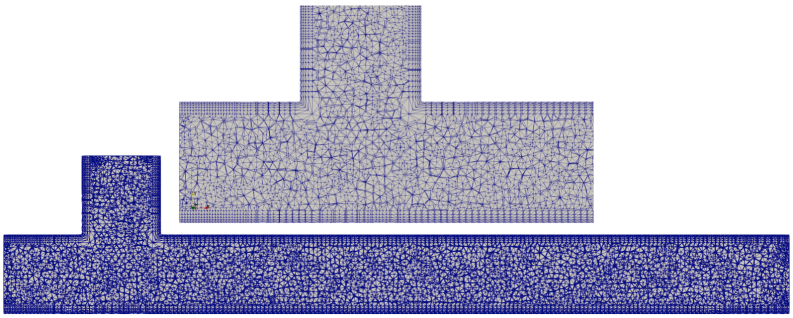

Figure 1.4: Lateral cut of an example of the simScale mesh

In addition, we have defined equal distances for the layers, near the walls, which will be modified later, if needed, to improve the bubble generation behavior. The figures 1.3 and 1.4 show the SimScale mesh, and the figures 1.5 and 1.6 show an example of the ICEM meshing for the T-junction.

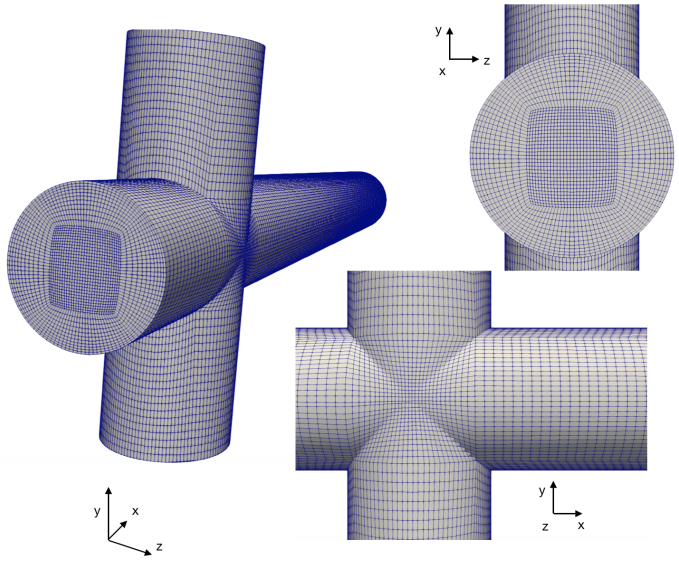

Figure 1.5: 3D representation of an example of the ICEM mesh

For the simulations, we had to define the boundary conditions, which needed the geometry to be separated into different parts. For the T-junction, which can be seen in the figure 1.7, we had the inlet1 for air, the inlet2 for water, the outlet for the resulting flow, and the walls 1 and 2. As for the double T-junction (figure 1.8), it was separated into the inlet1 for air, the inlets 2 and 3 for water, the outlet, the walls 1, 2, 3 and 4. **ParaView** was used to visualize each part of the geometry.

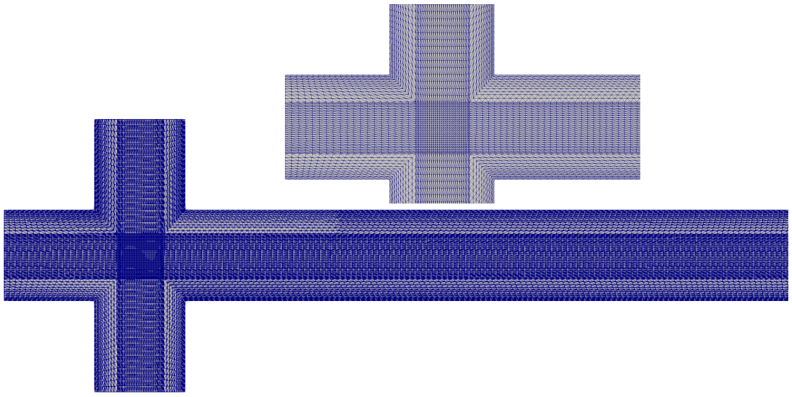

Figure 1.6: Lateral cut of an example of the ICEM mesh

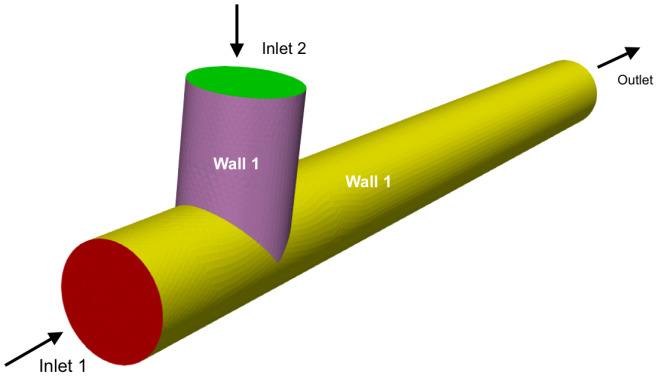

Figure 1.7: 3D representation of the different parts of the T-junction

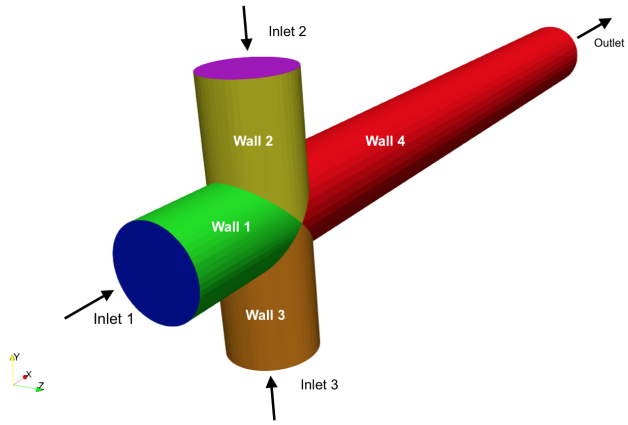

Figure 1.8: 3D representation of the different parts of the double T-junction

1.5. The problem statement

Our study is based on the CFD simulation of a two-phase flow composed of water and air, which will be considered incompressible, isothermal, and at a room temperature of 25°C. The water's dynamic viscosity and density are $\mu_L = 10^{-3}\ Pa$, $\rho_G = 10^3\ Kg/m^3$, respectively. The air has a dynamic viscosity $\mu_G = 10^{-5}\ Pa$, and a density $\rho_G = 1,225\ Kg/m^3$. We will work with a surface tension at the gas-liquid interface of $\sigma = 0,072\ N/m$.

	$U_{SL} - U_{SG}$ m/s	R_e	B_o	W_e	C_a
Minimum	0,318−0,059	377	0,139	4,01 10^{-3}	4,42 10^{-3}
Maximum	0,531−0,505	1036	0,139	3,00 10^{-2}	7,37 10^{-3}

Table 1.1: Characteristic values of the convergence tests

Furthermore, we will describe the main parameters that affect our simulation. We will also justify the study region with this parameters. More details can be found in the book of Arias [6]. The Reynolds number R_e number is considered a very determinant parameter for our study since it has a direct effect on predicting whether the flow will be turbulent or laminar. It is a dimensionless quantity defined as $R_e = \frac{\rho_L U_M L}{\mu_L}$, where U_M is the total of the superficial velocities of the fluids (water + air) with respect to the pipe, L is a characteristic linear dimension, but in a two-phase mixture inside a cylindrical pipe, we use the diameter $\phi_C = 0,001\ m$. The Bond number B_o represents the balance between gravitational and surface tension forces. Gravity, in this study, plays a smaller role than the surface tension in the bubble generation process. As a consequence, it is defined as $B_o = \frac{\rho_L\, g\, \phi_C^2}{\sigma}$, where g is the gravity. The Weber number W_e also named the bubble velocity, evaluates the competition between the capillary and the liquid drag forces that, as a consequence, provokes the formation and detachment of bubbles. It is defined as $W_e = \frac{\rho_G\, U_B^2\, \phi_C}{\sigma}$ Where U_B is the gas velocity for the double T-junction. Finally, the capillary number C_a is used to compare the viscosity and surface tension effects at the air-water interface. It is defined as follows $C_a = \frac{\mu_L\, U_{SL}}{\sigma}$, where U_{SL} is the water superficial velocity. The table 1.1 contains the maximum and minimum values of the R_e, B_o, W_e, and the C_a, in order to define the work region. The interval of R_e proves the laminarity of our study ($R_e < 2.300$). In addition, we can see the gravity independence ($g = 0\ m/s^2$) from B_o, which have been obtained experimentally as mentioned in the previous work of Arias [3]. From the value of W_e, the capillary forces are stronger than the inertial forces ($W_e < 2$). Finally, the surface tension effects are stronger than the viscosity ($C_a < 10^{-2}$).

1.6. Convective Time Units

We have checked some parameters to make sure that the simulations are correctly defined. The concept of the Convective Time Units (T_C) is the ratio between the product of the simulation time (in seconds) and the inlet velocity, and the pipe length. From our OpenFOAM case, if we compute 1 second of simulation time, we have approximately

computed 100 convective time units. Normally, this is good enough for a normal parametric variation project. Computing it for our combination 1 of velocities since it is the most extreme case. We have found the following values:

$$T_{C_G} = \frac{5\ 1,091}{0,01} = 545\ time\ units \tag{1.7}$$

$$T_{C_L} = \frac{5\ 0,160}{0,01} = 80\ time\ units \tag{1.8}$$

We can see that the liquid convective units are the threshold, therefore if we run a simulation for 5 seconds, it is like 80-time units in general. In other words, if we take one liquid particle at the inlet with a velocity of 0.16 m/sec, to travel all along the length of pipe (10 mm) to the outlet, it needs 0.0625 seconds. So, if we run the simulation for 0.0625 seconds, we make this liquid particle to move from the left to the right end of the pipe. In 5 seconds of simulation time, we can send 80 particles one after another. Launching a new particle at the inlet only happens when the previous particle reaches the outlet.

1.7. The boundary and initial conditions

The initial and boundary conditions must be carefully defined in order to solve the mathematical equations. The conditions for pressure, velocity and alpha fields are defined in the "0" folder of OpenFOAM. We have defined these conditions for the pressure, velocity and air fraction in the different parts of the T-junction (the inlets, outlet, and walls).

1.7.1. The initial conditions

For the velocity file, we have defined different combinations of uniform superficial velocities for the air and water in the inlets. The vector of the liquid superficial velocity U_{SL} is perpendicular to the circular cross-section of the vertical pipe, and the vector of the gas superficial velocity U_{SG} is perpendicular to the circular cross-section of the horizontal pipe. For the air fraction file, we have initially defined $\alpha = 1$ for the horizontal pipe, which is for air, and $\alpha = 0$ for the vertical one, which is for water.

1.7.2. The boundary conditions

For the velocity file, we have used the "fixedValue" condition for both inlets in order to define the values of U_{SL} and U_{SG}, and the same for the walls by using zero velocities. We have applied the "inletOutlet" condition for the outlet, which is similar to the "zeroGradient condition" in terms of not forcing any value, but it adds the condition of stopping the fluid from moving backward. For the pressure file, we have worked with the zero gradient condition for both inlets and walls, and we have fixed the atmospheric value for the outlet with the "fixedValue" condition. Finally, for the air fraction file, we

have used the "zeroGradient" condition for the walls and the outlet since the values will be changing during the simulations.

The contact angle is an important boundary conditions that deserves to be explained separately. It measures the angle between the liquid-gas interface when it meets a solid surface. It is a result of the interfacial tension between the liquid and the solid (walls). The interfacial tension is high on a hydrophilic surface but low on a hydrophobic one. We can see a representation of the contact angle in the figure 1.9.

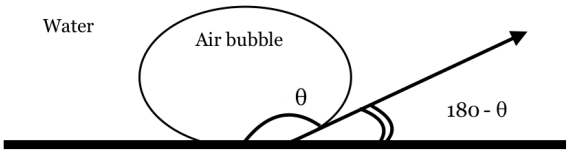

Figure 1.9: Representation of the contact angle between the solid wall and the bubble

We will add this condition in the air fraction file where the gas and liquid boundary and initial conditions have been defined. We will use the "constantAlphaContactAngle" command since we will operate with a fixed angle and not a dynamic one during the simulations. In addition, in OpenFOAM we define $180 - \theta$ and not the contact angle θ. For example, if we put 60º, the program will understand that the contact angle is 120º.

Observation: Some of the boundary conditions will change in order to improve the bubble generation quality in the chapter of the validations. In appendix A, the final boundary conditions, used for the chapter of results, can be found.

CHAPTER 2. THE VALIDATIONS

In this chapter, we will explain the convergence tests made in order to find the most suitable mesh when it comes to the number of cells as well as a good time step for to the simulations. However, this desired conditions will be limited by the real simulation clock time and the computers' capability. In addition, once we have the chosen mesh, more tests will be done in order to enhance the quality of the bubbles in terms of its shape, generation frequency, etc. Finally, the double T-junction will be introduced, described and tested in order to be used in the chapter of the results.

2.1. The convergence tests

In this section, numerical simulations have been run in order to test different meshes in order to find the most suitable one for the chapter of the results. At first, we wanted to define three time-steps and three meshes, from which we would choose the correct values that would be used as a reference to calculate the errors. The time steps $2,5 \times 10^{-7}\ s$, $5 \times 10^{-7}\ s$, and $10 \times 10^{-7}\ s$ were going to be tested, as well as the SimScale meshes of 95.500, 300.400 and 800.00 grids because these values resemble the ones used in the article of Arias [4]. However, we had to make changes due to two main reasons. Before mentioning the reasons, we will first describe the Courant number, which is directly related to the cell-size and time step as well as the fluid velocity. A physical explanation could be that C_o tells us how the fluid moves through our computational cells. Robust systems as OpenFOAM can easily deal with large C_o numbers (hundreds) but not for all cases since it can cause convergence problems and it can also affect the accuracy of a transient simulation. It is defined as $C_o = \frac{U\,\Delta t}{\Delta x}$. Consequently, the unstructured meshes created with simScale kept causing the sudden increase of the Courant Number due to some extremely small cells, which have constantly lead to the abortion of the simulations. Second, the available machines needed higher values of Δt to fasten the computations, but not too high since it is directly related to C_o. As a result of both impediments, we have ended up using new meshes, created with ICEM as mentioned in the previous chapter, of 220.000, 400.000 and 600.000 grids, which have a controllable cell sizes. As for the time step, we used a bigger value, that of $\Delta t = 5 \times 10^{-6}\ s$ to fasten the simulations.

Finally, for the combinations of gas and liquid superficial velocities for the convergence tests. The following diagram in the figure 2.1 presents the six combinations, which, at first were four. However, two combinations could not lead to the convergence of the simulations as it will be explained later. As a result, we have added two new combinations since we need, at least, four valid cases to well-define the physical region where the study is carried out. As for the values of their values, it is important to check the ranges and cover all the main cases, which are: $U_{SL} < U_{SG}$, $U_{SL} > U_{SG}$, and finally similar values of U_{SL} and U_{SG}. Furthermore, we have selected these exact values since there are already existing experimental data that can be used as a reference not only to monitor our bubble behavior but also for possible comparisons between the bubble parameters.

With these simulations we will calculate the bubble frequency f_B, the bubble velocity U_B, the bubble volume V_B and the bubble length L_B, and then we will calculate the

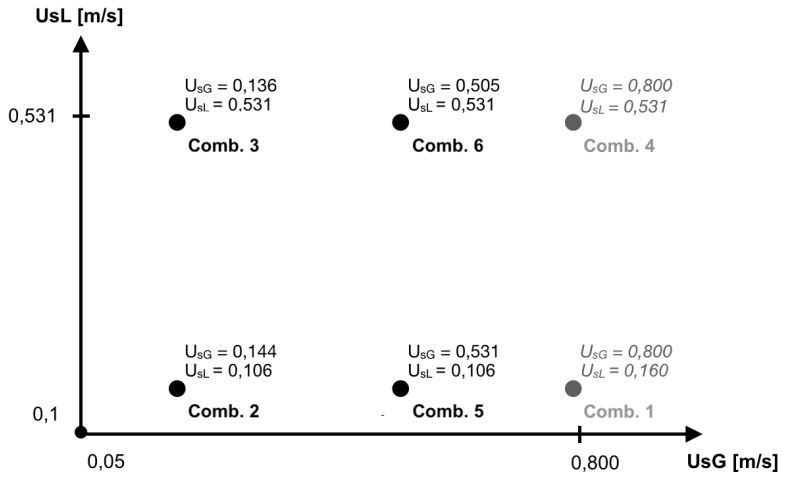

Figure 2.1: The air and water superficial velocities of the convergence tests

errors. We have used the mesh of 600.000 grids as a reference to compute these errors, since it is the most refined one among the three we have created.

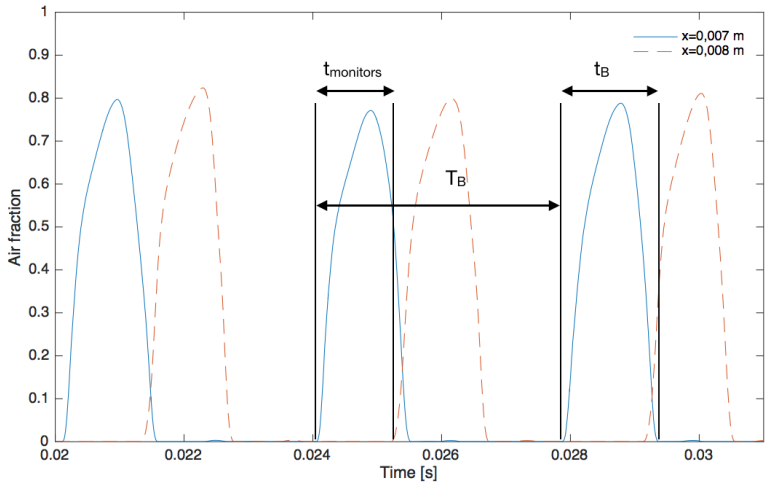

Figure 2.2: The bubble generation detection in the monitors placed at x=7mm and x=8mm

From the simulations we have obtained a file containing the values of void fraction α and the time step. We have used **matlab** to generate the plots from which we obtained the desired parameters mentioned previously. The figure 2.2 shows a matlab plot of the bubbles that will be used to understand the following formulas of the bubble parameters:

$$f_B = \frac{1}{T_B} \tag{2.1}$$

where T_B is the bubble generation period.

$$U_B = \frac{d_{monitors}}{t_{monitors}} \tag{2.2}$$

where $d_{monitors}$ is the distance between the monitors placed at $x = 0,007\ m$ and $x = 0,008\ m$ as in the articles of Arias [4] [3], and $t_{monitors}$ is the time that the bubble needs to travel from the first monitor to the second one.

$$L_B = U_B\, t_B \tag{2.3}$$

where t_B is the bubble duration.

For every simulation, we have saved a certain number of bubbles depending on when the bubble generation has become stable. This number varied from one simulation to another. Some simulations have reached stability after 16 generated bubbles, and some others after 90. We have computed the mean values of the parameters for the last five bubbles of the total number. In addition, we have computed the standard deviation σ_B in order to add credibility to our calculated errors.

$$\sigma_B = \sqrt{\frac{1}{1-B}\sum_{i=1}^{B}(x_i - \overline{x})^2} \tag{2.4}$$

where B is the number of bubbles, in this case B=5, x_i is the value of each parameter and $\overline{x}$ is the mean value of the five bubbles.

2.1.1. The pre-tests for the combinations 1 and 4

For the combinations 1 and 4, the difference in the air and water superficial velocities is the main cause for the gas attachment in the walls and the non-generation of bubbles. As a result, we have added the Contact Angled condition in order to obtain bubbles.

In this case, we have tested different values of the contact angle in order to find the most suitable one for our case. In addition, we have stuck to the condition of equal horizontal and vertical angle ($\theta_H = \theta_V$). Also, we have done this test with the most refined mesh (600.000 grids). In the next subsections, we can see the results of α, f_B, U_B and L_B for different values of θ.

Combination 4

In table 2.1, it can be seen that all the tested angles give the same results. It means that the contact angle condition couldn't have any effect on the bubble generation. In addition, the bubble velocity is reasonable since it is higher than the gas superficial velocity. As a consequence, we will work with the angle 25°, which is neither big nor small.

Figure 2.3: The bubble generation of the combination 4 pre-test for θ=25°

Angle	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (mm)	σ_{L_B} (mm)
θ=10º	432,65	84,81	0,8553	0,7321	1,7025	0,7111
θ=25º	432,65	84,81	0,8553	0,7321	1,7025	0,7111
θ=40º	432,65	84,81	0,8553	0,7321	1,7025	0,7111

Table 2.1: The pre-tests of the combination 4 for different contact angles

To conclude, none of these angles give optimal results, which means that the correct angle for this combination of velocities is not valid. However, we have decided to stick to the angle of 25º because it has given the most acceptable values even if the bubble generation was very poor (see Figure 2.3).

Combination 1

The superficial velocities $U_{SL} = 0,160\ m/s$ and $U_{SG} = 0,800\ m/s$ are the most difficult case to simulate. The bubble generation is unstable. Moreover, the gas often occupies the horizontal pipe completely, which prevents the computation of the bubble parameters with matlab. As a result, we have, again, tried different contact angles. We have tried the contact angles 10º, 25º, 30º and 40º. In addition, instead of 600.000 grids for the third mesh, we used 500.000 grids more or less, since the simulations could not converge with a higher number of grids due to the Courant number.

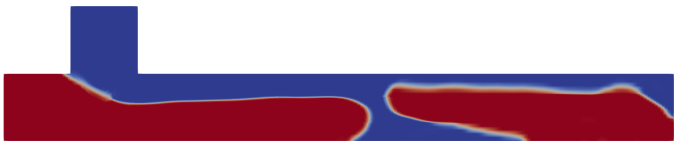

Figure 2.4: The bubble generation of the combination 1 pre-test for θ=30º

Angle	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (mm)	σ_{L_B} (mm)
θ=25º	98,05	72,052	0,0148	0,0045	0,2403	0,2133
θ=30º	120,02	50,935	0,0434	0,0030	0,4968	0,1443
θ=40º	37,78	0,537	0,5543	0,9441	0,5011	0,2254

Table 2.2: The pre-tests of the combination 1 for different contact angles

The table 2.2 shows the bubble results for the angles 25º, 30º and 40º. The bubble generation frequency of 40º is very low compared with the rest of the angles. For the angles 25º and 30º, the bubble velocities are very low, but the bubble frequencies are high and similar. As a result, we have decided to use the angle of 30º to calculate the errors of the meshes, since it has given a higher frequency. The figure 2.4 presents the bubble generation for the angle 30º, and we can see that the bubbles are very long with a bad shape.

2.1.2. The Errors and the standard deviation

Finally, the results of the tests can be seen in the table 2.3, which en-globes all the errors and standard deviations for all the combinations of velocities. We can also see that the third mesh has no errors because, as mentioned previously, it is the one used as a reference. We can see that the errors are very high for the combination 4 due to the difference between U_{SL} and U_{SG}. In addition, the standard deviations show the high fluctuations of the bubble parameters, which are not able to converge. As for the combination 1, we could not obtain any results. Both combinations of velocities are not valid. The combinations 2 and 3 have the best error results, especially the combination 3. The frequency's standard deviation of the combination 2 is lower, which is good, but the frequency values of the combinations 3 are higher so the deviation is not really significant. For the extra combinations of velocities 5 and 6, the results are good as it can be seen in the same table, and their standard deviations are also very small. As a consequence, these two combinations are valid.

U_{SL}-U_{SG}	f_B (Hz)	σ_{f_B} (Hz)	ϵ %	U_B (m/s)	σ_{U_B} (m/s)	ϵ %	L_B (mm)	σ_{L_B} (mm)	ϵ %
0,106−0,144	86,97	0,580	3,74	0,254	5,307 10^{-4}	1,11	1,829	0,0134	5,97
	89,04	0,548	1,45	0,257	0,0013	2,11	1,812	0,0135	4,97
	90,35	0,954	-	0,251	1.413 10^{-4}	-	1,726	0,0187	-
0,531−0,136	298,19	4,015	2,48	0,708	0,0061	2,11	1,016	0,0206	1,88
	284,02	9,559	2,39	0,714	0,0040	1,28	1,020	0,0121	2,24
	290,99	6,637	-	0,724	0,0214	-	0,998	0,0260	-
0,531−0,800	110,54	73,074	74,45	0,012	0,0015	98,63	0.177	0,1749	89,60
	292,93	139,195	32,29	0,512	0,8185	40,14	0,731	2,7932	57,04
	432,65	84,810	-	0,855	0,7321	-	1,703	0,7111	-
0,106−0,531	157,65	7,052	3,86	0,616	0,0029	0,77	3,393	0,1638	2,78
	155,16	3,209	2,22	0,615	0,0026	0,56	3,455	0,0688	1,01
	151,79	2,434	-	0,612	0,0026	-	3,491	0,0688	-
0,531−0,505	432,89	14,317	1,76	1,041	0,0977	0,11	1,654	0,1566	7,28
	430,88	4,426	2,22	1.047	0,0068	0,76	1,668	0,0889	6,51
	440,66	19,517	-	1,039	0,0602	-	1,784	0,2767	-

Table 2.3: The f_B, U_B and L_B errors and standard deviations for the all combinations of $U_{SL} - U_{SG}$ for the meshes of 220.000, 400.000 and 600.000 cells

In general and for all the combinations of velocities, the 400.000 grids mesh shows small errors and standard deviation values in comparison with the 220.000 grids mesh. However, in many cases, we see that the mesh of 400.000 cells gives even fewer fluctuations than the mesh of 600.000 grids, which should be the best option since it is the most refined mesh, used as a reference. Our four combinations 2, 3, 5 and 6 can be accepted as valid because they have given good results, and we can use the mesh of 400.000 grids in the next chapter of results. However, it is still not clear why this chosen mesh is, sometimes, better than the reference mesh. First, it could be due to the combinations of superficial velocities because, as it can be seen in the tables 2.3 and 2.4, for the the combination 6 where the gas and liquid have similar superficial velocities but high values,

the fluctuations of the 400.000 grids mesh are lower than the ones of the reference mesh. For the combination 2 where the gas and liquid have low similar superficial velocities, the fluctuations of the velocity are lower for the reference mesh but the opposite for the bubble frequency and length. In the case of the combination 3 that has a faster liquid, the fluctuations of the bubble frequency are lower for the reference mesh but the opposite for the bubble velocity and length. For the combinations 4 and 5 where the liquid is slower than the gas, we obtain the expected result of lower fluctuations for the 600.000 cells mesh. Second, it could also be due to the OpenFOAM solvers (PCG, smoothSolver, etc.) mentioned in the *fvSolution* file, on any other ambiguous factor of this file. The OpenFOAM program is a potent CFD simulator that we do not fully understand yet.

Parameter	Comb. 2	Comb. 3	Comb. 4	Comb. 5	Comb. 6
f_B	X	O	O	O	X
U_B	O	X	O	O	X
L_B	X	X	O	O	X

Table 2.4: Summary of the standard deviations results for the bubble parameters for all the combinations of velocities. X=The fluctuations of the 400.000 grids mesh are higher than those of the 600.000 grids mesh. O=Otherwise

From the figure 2.5, the bubble frequency and length curves for the 400.000 mesh are closer to the one of the 600.000 mesh. In the case of the velocity, It is not that regular because, at the beginning of the simulations, the mesh of 400.000 cells is more similar to the reference mesh but, then, the mesh of 220.00 grids becomes better. From the figure 2.6 and as explained in the previous paragraph, the 400.000 grids mesh shows more stability, but the curves of the bubble length for the 220.000 grids mesh are closer to the reference mesh. The bubble frequency and velocity, however, are the opposite. From the figure 2.7, the curves are chaotic, but, at least, the 400.000 grids mesh simulations show fewer fluctuations when compared to the other meshes. The same goes for the combination 6 as it can be noticed in the figure 2.8. The curves of the 400.000 cells mesh are the most stable. From the figure 2.7, which shows the results of the combination 5, the curves of the 400.000 grids for all the bubble parameters are more similar to the reference mesh than the case of the 200.000 grids mesh. These figures, once again demonstrate the suitability of the 400.000 cells mesh for the next simulations.

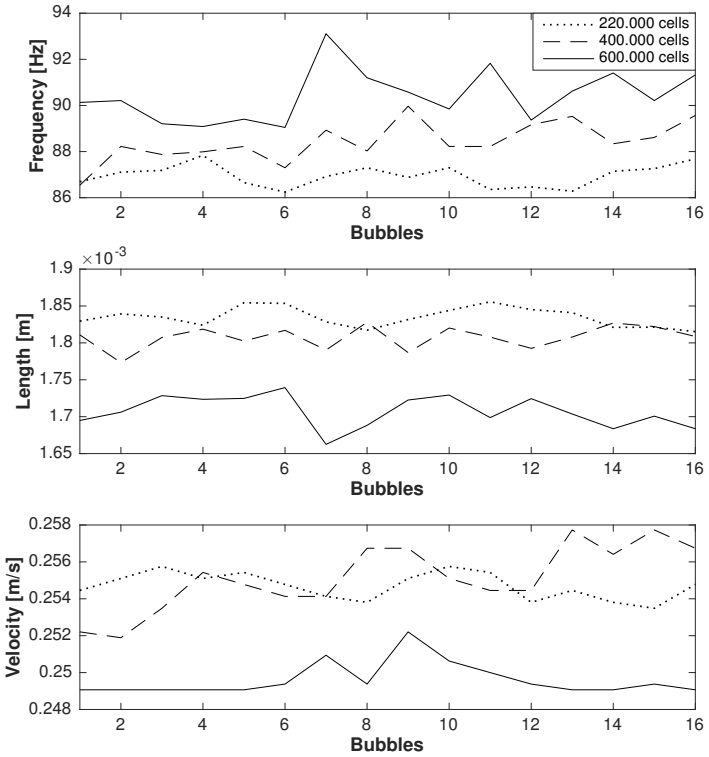

Figure 2.5: The f_B, U_B and L_B curves for the combination 2

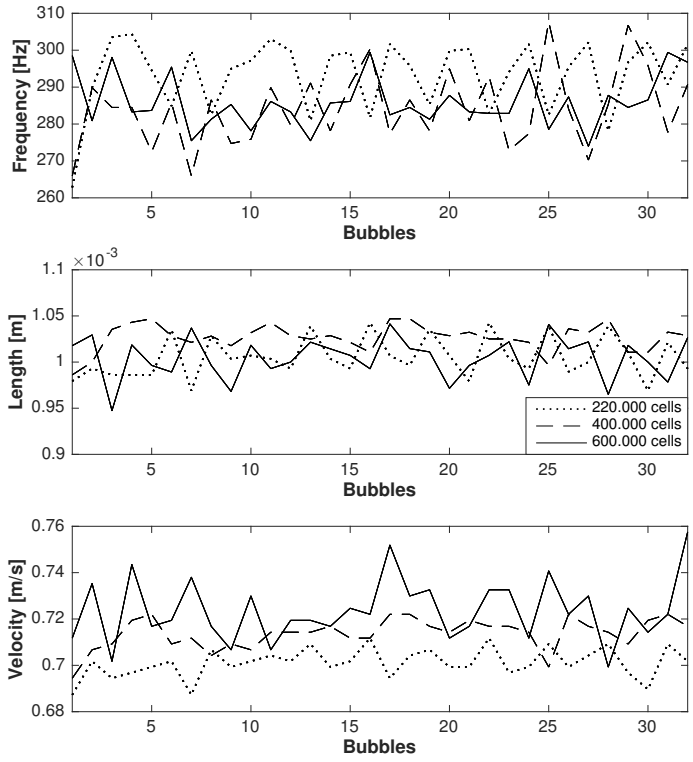

Figure 2.6: The f_B, U_B and L_B curves for the combination 3

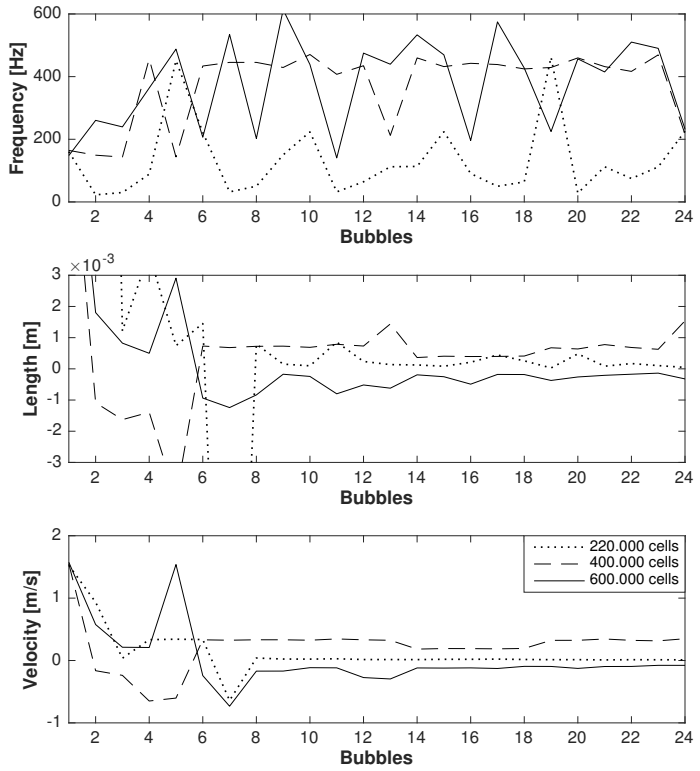

Figure 2.7: The f_B, U_B and L_B curves for the combination 4 with $\theta = 25°$

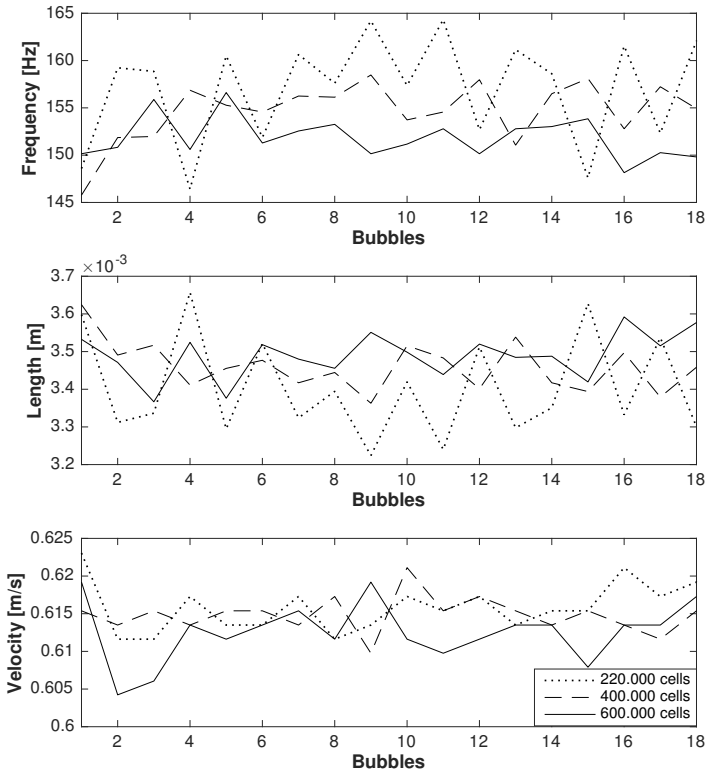

Figure 2.8: The f_B, U_B and L_B curves for the combination 5

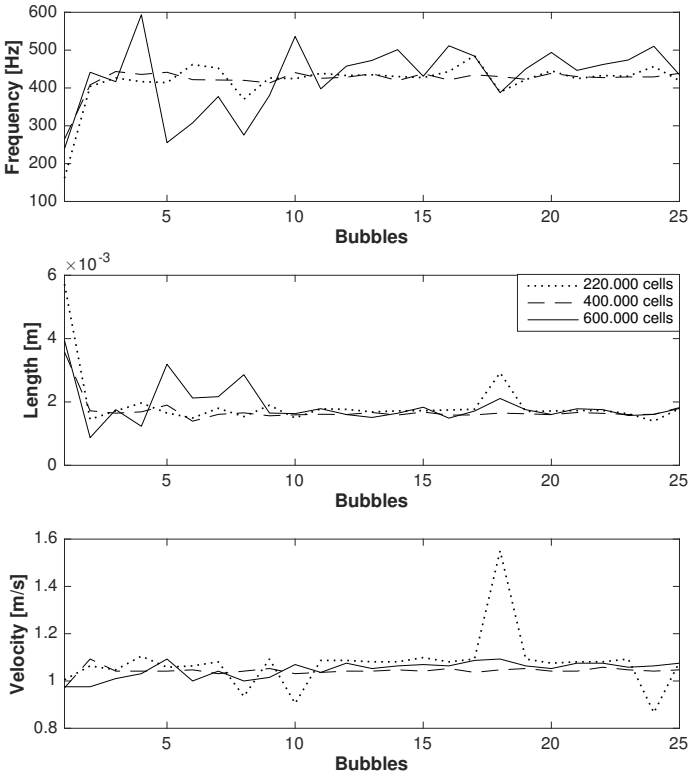

Figure 2.9: The f_B, U_B and L_B curves for the combination 6

2.2. Bubble detachment from the walls

2.2.1. The contact angle

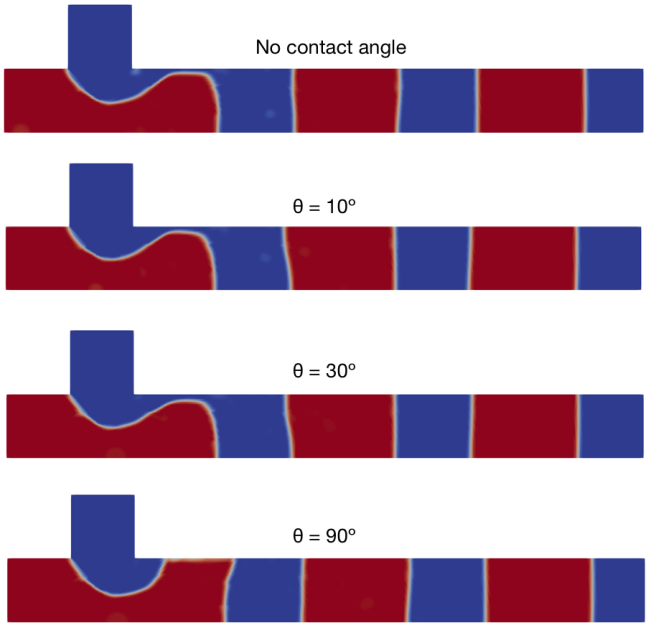

Figure 2.10: The contact angle tests for the combination 2

In this subsection, we will analyze the 400.000 grids mesh with different contact angles, in order to obtain the bubble detachment from the walls. We will simulate the angles 10°, 30° and 90° for the combination 2, and the angles 0, 30°, 30° and 90° for the combination 3. The contact angle tests presented in the tables 2.5 and 2.6 prove

Angle	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (mm)	σ_{L_B} (mm)
θ=10°	88,06	0,592	0,254	0,0007	1,807	0,0115
θ=30°	89,27	1,132	0,258	0,0012	1,824	0,0127
θ=90°	88,69	0,609	0,256	0,0014	1,811	0,0132

Table 2.5: The bubble generation of the contact angle tests for the combination 2

that the contact angle condition is not enough. For the combination 2, the bubble parameters have stable values but without solving the bubble attachment to the walls. As for the combination 3, the bubbles have been detached, yet with an unstable behavior. The figure 2.10, shows some **Paraview** images for the combination 2, where it can be observed that this condition has not prevented the bubble attachment to the walls, and the figure 2.11 shows the contact angle tests for the combination 3, where we can see that the bubbles have been detached for the angle 30°, but with a very unstable generation as well as gas dispersion. Consequently, the contact angle cannot be considered a good main solution for the bubble attachment to the walls.

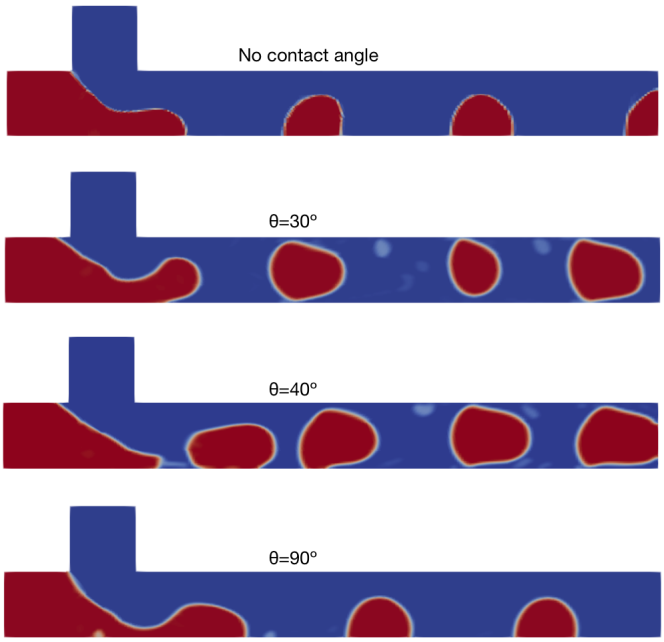

Figure 2.11: The bubble generation of the contact angle tests for the combination 3

Angle	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (mm)	σ_{L_B} (mm)
θ=0°	354,29	79,80	0,938	0,207	1,497	0,468
θ=30°	319,29	204,88	0,507	0,736	1,352	2,309
θ=40°	473,49	70,13	0,626	0,391	1,132	0,701
θ=90°	291,97	7,928	0,716	0,004	1,026	0,014

Table 2.6: The bubble generation of the contact angle tests for the combination 3

2.2.2. The zero gradient condition

In this case, we have tried to solve the issue by removing the non-slip condition of the velocity in the walls and putting the "zeroGradient" condition. Figure 2.12(a) show the result. The gas could not circulate with its given velocity, thus it has stayed in the inlet during the whole simulation. As a consequence, this option has been discarded

2.2.3. The Slip condition

We have implemented the opposite the non-slip condition which is the slip condition which allows the fluid to smoothly move in the walls with its corresponding velocity. The figure 2.12(b) shows how some of the bubbles have been able to detach from the lower wall but then stick to the upper one, irregularly. In addition, we have tried the same condition but with a contact angle of 70° (2.12(c)), which have not enhanced the bubble generation. Both solutions have also been discarded.

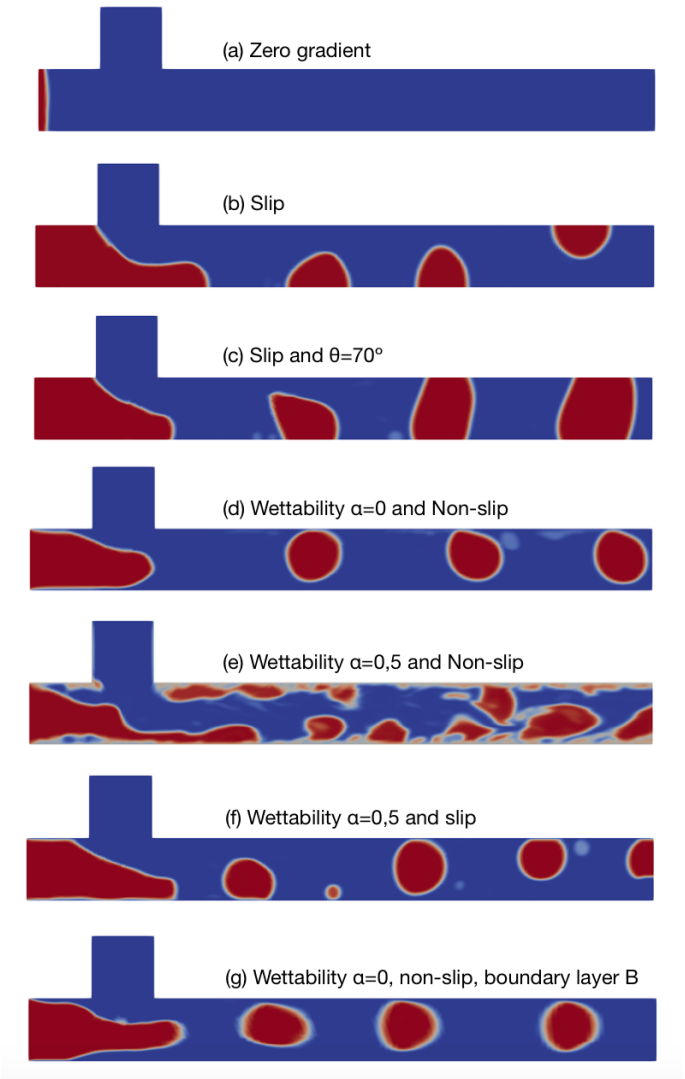

Figure 2.12: The different conditions for the bubble detachment from the walls for the combination 3

2.2.4. The wettability condition

Next, we have implemented the wettability condition, which consists in imposing the walls to always have water. We have changed the zero gradient condition of the walls for the air fraction into α=0 as well as α=0,5. The figures 2.12(d) and 2.12(e) show that these two conditions have caused important changes. However, only the α=0 has almost given the desired results. The bubbles are close to the upper wall, which causes the loss of material ass well as the decrease of the bubble velocity U_B. As a result, This condition needed more tests and modifications. Also, we have implemented the wettability and the slip conditions at the same time. The figure 2.12(f) shows bad results, which means that the slip condition is not acceptable in any case.

2.2.5. The boundary layer

Mesh	Boundary layer	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (m)	σ_{L_B} (m)
400.000 cells	9 layers (A)	301,25	147,0348	0,789	0,0513	1,124	0,0799
450.000 cells	**14 layers (B)**	357,66	42,4698	0,839	0,0098	1,022	0,0157
470.000 cells	9 layers (C)	269,42	79,8587	0,8136	0,0515	1,029	0,1284
460.000 cells	9 layers (D)	330,74	116,4905 0,776	0,0210	1,039	0,1069	

Table 2.7: The bubble parameters for different boundary layers for the combination 3

Finally, we have combined the wettability condition with different boundary layers in new meshes of 400.00 grids approximately. A balance between the number of cells in the pipes and the number of layers has been made in order to keep the overall number of grids around 400.000 of the chosen mesh. The table 2.7 sums up the 4 meshes that have been generated for this tests with their bubble results and standard deviations. As it can be seen from the standard deviations, the boundary layer B is the most suitable one for our chosen mesh. The figure 2.12(g) show the bubble generation for this final mesh.

2.2.6. The optimal solution

Now that we have obtained the bubble detachment from the walls, we will add the bubble volume V_B as a new parameter.

$$V_B = \int_{t_a}^{t_b} \alpha \, dt \, U_B \, (\pi \, \frac{\phi_C^2}{4}) = A_B \, U_B \, (\pi \, \frac{\phi_C^2}{4}) \tag{2.5}$$

where t_a and t_b are the instants in which the bubble starts and ends, respectively.

We have computed the bubble parameters and the standard deviation for the acceptable possible solutions which are the ones with the wettability condition, in order to compare them and select the most suitable one to be used the chapter of the results. The table 2.8 shows the obtained values. We have also compared these results with the experimental case, which has given the following bubble parameters: $f_B = 185,8 \, Hz$, $U_B = 0513 \, m/s$, $L_B = 2,08 \, mm$. We can see that our results were not similar to the experiments. However, it is acceptable as it is difficult to meet the same conditions since, for example, we design and use different meshes which do not exist in the real experiments. The figure 2.13 shows the experimental bubble generation compared with our simulations.
Finally, we have added the contact angle condition in order to obtain better results, more similar to the experimental case of the combination 3. We have used the mesh with the boundary layer B of 14 layers, which gave us the lowest standard deviations, in addition to the wettability condition ($\alpha = 0$). The results have been compared to the simulation that has no contact angle. From the table 2.9, it can be seen that the results are different after adding θ. It is mainly to the activation of the interfacial solid-gas tension, which is directly related to angle. Furthermore, the different angles have no

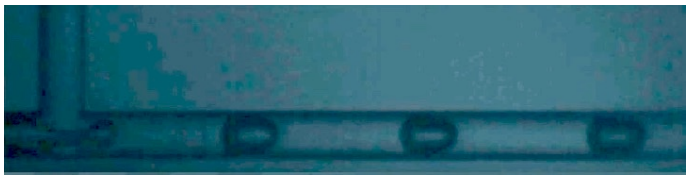

Figure 2.13: The experimental bubble generation for the combination 3

Boundary condition	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (m)	σ_{L_B} (m)
Wett. + Slip	409,55	249,047	0,845	0,235	0,892	3,149
Wett. + No-slip	335,66	146,451	0,089	0,012	0,229	0,106
Wett. + No-slip + Boundary layer B	357,66	42,469	0,839	0,009	1,022	0,016

Table 2.8: The bubble parameters comparison for for the combination 3

Contact angle	f_B	σ_{f_B}	U_B	σ_{U_B}	L_B	σ_{L_B}	$V_B\,10^{-10}$ (m^3)	$\sigma_{V_B}\,10^{-10}$ (m^3)
$\theta = 0^o$	349,82	37,614	0,831	0,007	1,022	0,018	3,149	0,116
$\theta = 45^o$	349,82	37,614	0,831	0,007	1,022	0,018	3,149	0,116
$\theta = 90^o$	349,82	37,614	0,831	0,007	1,022	0,018	3,149	0,116

Table 2.9: The bubble parameters and their standard deviations for different angles for the combination 3. Comparison with $f_B = 357,6604\ Hz$, $U_B = 0,8397\ m/s$, $L_B = 1,0217\ mm$, $V_B = 76,4760\ 10^{-10}\ m^3$

effect in the bubble results, mainly because of the gas detachment from the wall. The different contact angles have the same values because the wettability conditions stop the gas from touching the walls, which makes the contact angle ineffective.

2.3. The double T-junction

In this final section, we will test the double T-junction in this last section of the validations chapter. The superficial velocities of the water, if the T-junction has U_{SL}, the vertical inlets of the double T-junction will have $U_{SL}\ /\ 2$ each one. Since we have selected the mesh of approximately 400.000 cells, and a boundary layer of 14 layers (Boundary layer B) for the T-junction. We have employed a mesh of nearly 460.000 cells for the double T-junction. Also, since this geometry has an extra water inlet, it will cause the increase of the grids number if the same number of 14 layers is used. Therefore, a boundary layer of 11 layers has been decided on.

We have simulated the combination 3 of superficial velocities ($U_{SL} = 0,531\ m/s$ and $U_{SG} = 0,136\ m/s$) for the following tests:

- **Test 1:** We have used the same boundary conditions that have been chosen for the T-junction in the previous section (wettability condition $\alpha = 0$ in the walls).

Condition	f_B (Hz)	σ_{f_B} (Hz)	U_B (m/s)	σ_{U_B} (m/s)	L_B (mm)	σ_{L_B} (mm)	$V_B\ 10^{-10}$	$\sigma_{V_B}\ 10^{-10}$
Test 1	216,88	37,277	0,840	0,024	1,295	0,104	5,163	0,443
Text 2	258,01	1,749	0,806	0,014	1,207	0,092	4,931	0,074
Test 3	232,94	45,516	0,836	0,035	1,272	0,092	5,034	0,493
Test 4 (0°)	211,74	5,231	0,819	0,016	1,299	0,038	5,359	0,263
Test 4 (45°)	249,65	9,208	0,841	0,028	1,248	0,077	5,125	0,344
Test 5 (60°)	141,34	0,282	0,709	0,007	1,055	0,004	4,092	0,475
Test 6	206,03	3,917	0,719	0,011	1,196	0,057	5,385	0,104
Test 7	235,64	23,768	0,850	0,022	1,273	0,023	5,017	1,555

Table 2.10: The bubble parameters and their standard deviations for the different tests of the double T-junction for the combination 3 (The volume unit was deleted so that the table could fit in the page)

- **Test 2:** We have used the contact angle condition of $\theta = 45°$ in all the walls without the wettability condition.
- **Test 3:** We have tried the same situation but with a contact angle $\theta = 45°$ in all the walls. Thanks to the wettability condition, the bubbles have been detached from the walls. Nevertheless, it, at the same time, interferes with the contact angle condition that needs to be activated correctly in the pipes before reaching the wall 4, or two boundary conditions in OpenFOAM might not work properly if activated at the same time.
- **Test 4:** As a result of the previous simulation, we have modified it so that the contact angles $\theta = 0°$ and $\theta = 45°$ have been activated for the walls 1, 2 and 3, and the wettability condition $\alpha = 0$ for the wall 4.
- **Tests 5:** We have tried the contact angle 90°, 80°, 70° and 60° for the split conditions. It has only worked for the last angle.
- **Test 6:** The gas could not flow for contact angles more or less bigger than 60°, therefore, we have simulated the test 2 but with $\theta = 90°$, in order to see if the cause of the problem is the separation of the boundary conditions of contact angle and wettability (Test 4).
- **Test 7:** We have also simulated the test 3 but with $\theta = 90°$.

From the figure 2.14 we can see that the bubble length is bigger for $\theta = 90°$ than for $\theta = 45°$. This behavior is correct since, for the numerical results of the T-junction [4], L_B increases with the increase of the contact angle. For the cases in the figure 2.15, it is difficult to compare the lengths. It seems that the bubble lengths for 0° and 45° are bigger than for 60°, which is not correct but we cannot assume it from these tests.

For the **test 1**, the wettability condition alone shows standard deviations bigger than the **test 2**, which has only the contact angle. Unlike the previous geometry, for the double T-junction, the contact angle was good enough to detach the bubbles, as well as giving very stable results. For the **test 3**, the fluctuations again were high, similar to

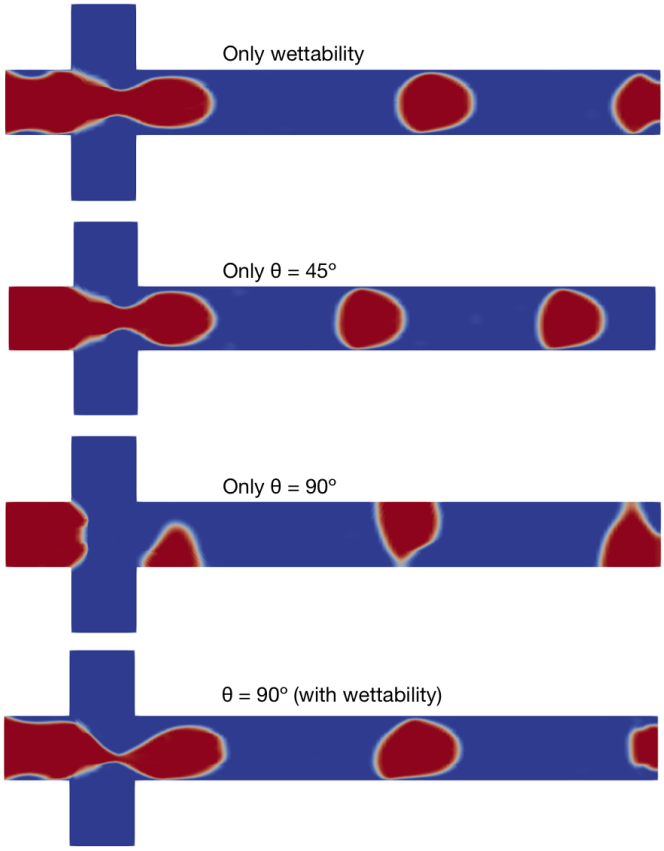

Figure 2.14: The bubble generation for the tests 1, 2, 6 and 7

the test 1. Consequently, the wettability condition, when imposed at the same time as the contact angle, its effects are stronger. For the **tests 4** and **5**, The contact angles 80°, 90°, 70°, and 60°, the simulation could not run properly except for the last angle. One explanation could be that, since the contact angle is propositional to the surface tension, for high values of θ, the strong surface tension stops the gas from flowing, because the walls become hydrophilic. Nevertheless, we did not have this problem with the T-junction while using big contact angles, but again, we also did not have the geometry split, with the wettability and contact angle conditions separated. It is not physically possible to use different boundary conditions for different parts of the real geometry because the material of all the pipes is the same. As a result, we have added the **tests 6** and **7**, where we have simulated $\theta = 90°$ with and without the wettability condition. In the figure 2.14 the angle 90° without the wettability condition does not work because the bubbles are being irregularly attracted to the walls. Also, the way the gas is cut is different from the rest of the cases, and the same happens to bubble generation for the $\theta = 60°$ of the test 5 as it can be seen in the figure 2.15.

The table 2.10 shows the bubble results and the standard deviations. The standard deviations of the frequency are higher for the tests 3 and 7, which have the wettability and contact angle applied at the same time. One explanation could be that both conditions are not compatible with each other. Also, the frequency for the angle 60° is very low compared with the rest, and the gas velocity is low as well as for the test 6. Consequently, we could say that for big angles, the bubbles become slow due to the

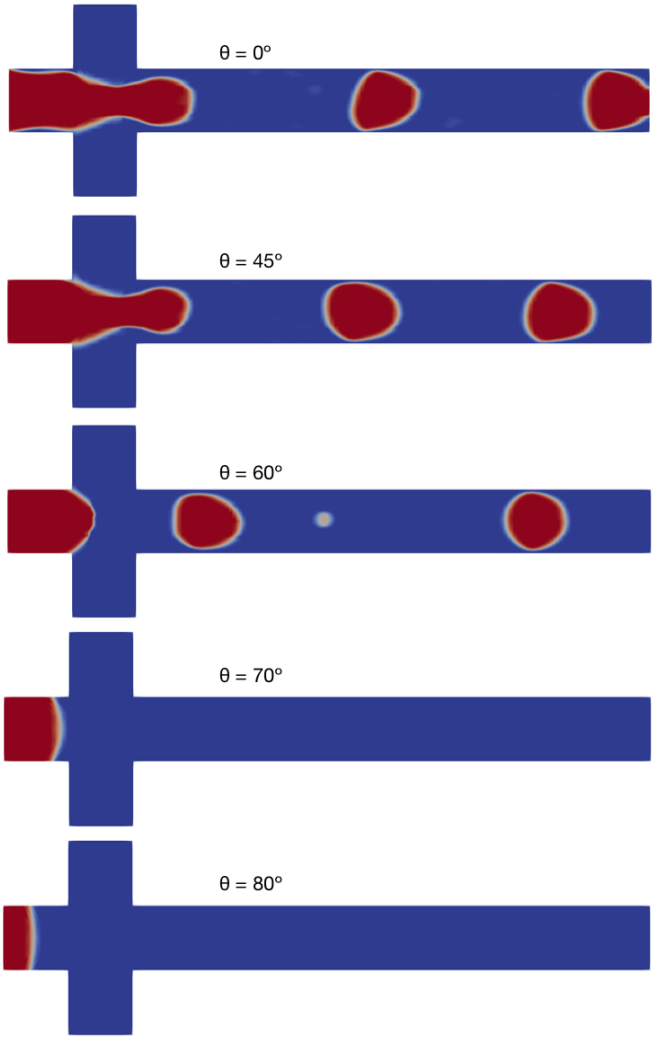

Figure 2.15: The bubble generation for the tests 4 and 5

surface tension, which is proportional to the contact angle. The similarity between both tests is that the big contact angle is applied alone in the beginning of the horizontal pipe where it has great effect on f_B and U_B.

One important thing to mention is that these results are only for the combination 3 of velocities. It might be different for the rest of combinations as it happened for the T-junction. Besides, the bubble detachment from the previous configuration, for the combination 3, was not optimal with the contact angle alone (30°) due to the fluctuations and gas dispersion. Lastly, one explanation of the bubbles being more stable and well detached for the double T-junction is that with two vertical water inlets it is easier for the gas to be cut to create an almost round bubble not attached to the walls. It might be because the upper and lower sides of the gas are receiving the same quantity of water force, unlike in the previous T-junction.

To sum up this section, it is more than clear that the contact angle still needs more tests. That is why we will work with the wettability condition in the chapter of results for both geometries that need to be compared under the same conditions.

CHAPTER 3. RESULTS AND DISCUSSION

In this chapter, we will finally analyze the results of the simulations that will be performed, for the T and double T-junction. We will compare the numerical results of both geometries, with new combinations of velocities, different from the ones of the convergence tests. We have chosen two groups of five combinations of velocities each. The first group has $U_{SL} = 0,318\ m/s$ and the second group has $U_{SL} = 0,531\ m/s$. Similarly to the combinations for the convergence tests, these superficial velocities have been chosen due to the already existing experiments performed with them. The tables B.1, B.2, B.3 and B.4 contain these 10 combinations of velocities.

Observation 1: Note that in the next sections and Appendix B, the sub-index "D" refers to the double T-junction, the "T" to the T-junction, and the "E" to the experiments. For the fitting curves in the plots, the solid line is for the experiments, the dashed line is for the T-junction, and the dotted line is for the double T-junction.

Observation 2: The figures 3.1, 3.2, 3.3, 3.4 and 3.5 show the results of the experiments, the T and the double T junctions. However, the detailed information about the experimental data and its comparison with the T-junction can be found on the work of Dalfó [10] and Arias [4] [3].

3.1. Bubble frequency

In this section, we will analyze the bubble generation frequency. The table B.1 contains the bubble generation frequencies, their standard deviations, and the errors. The figure 3.2 is a Matlab plot of f_B as a function of the different air superficial velocities U_{sG}. It shows how f_B has two phases. It first increases with the increase of the gas superficial velocities U_{SG}, then the second phase is when it reaches its saturation values for the two water superficial velocities in the experiments, T-junction and double T-junction. The frequency behavior can be fit with the following expression:

$$f_B = f_{sat}\ (1 - e^{-U_{SG}\ (a_0\ /\ f_{sat})}\ ) \tag{3.1}$$

Where a_0 is the slope of the linear first phase, and f_{sat} is the saturation frequency.

U_{SL} m/s	a_{0D}	a_{0T}	f_{satD} Hz	f_{satT} Hz
0,318	1136	1135	427,1	520,4
0,531	1928	5086	1261	480,1

Table 3.1: The slope and the saturation frequency for both geometries.

The table 3.1 sums up these parameters, which have been computed with the Matlab "fit" function, of the T and double T junctions for the two U_{SL}. The saturation frequency of the double T-junction for $U_{SL} = 0,531\ m/s$ does not appear in the plot because of

the axis limitation. Also, f_{sat} increases with the increase of U_{SL} as an outcome of the drag force, which is involved in the bubble detachment [4]. This explains the increase of a_0 with the increase of the liquid superficial velocity. However, for the T-junction, the bubble generation does not follow this rule. Consequently, it needs more research, and it can be found in the thesis of Dalfó [10], where a third group of superficial velocities with $U_{SL} = 0,106/s$ has been added.

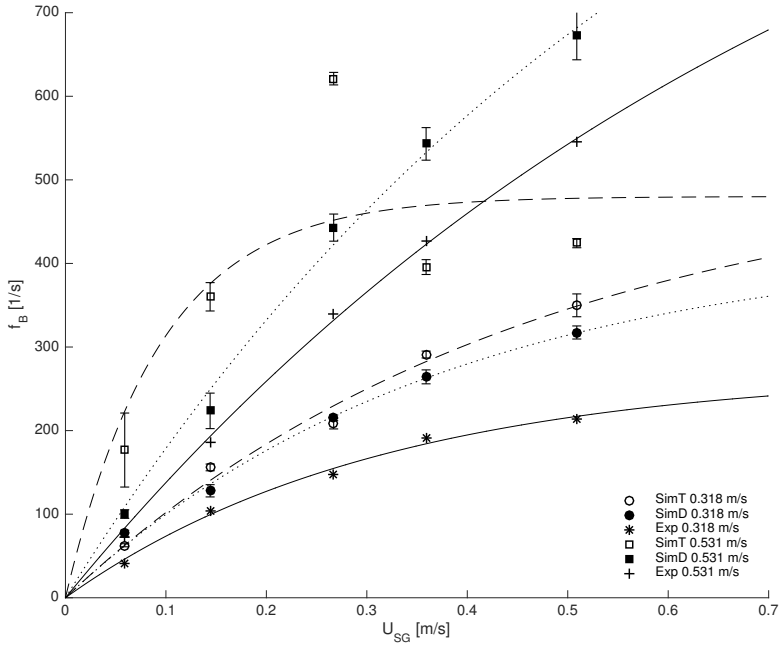

Figure 3.1: Bubble frequency as a function of the superficial gas velocity for two different superficial liquid velocities.

From the figure 3.1, the bubble generation frequency for the double T-junction increases with the increase of the gas and liquid superficial velocities, and its fitting curves agree qualitatively with the experiments. However, it can be seen in both the figure and the table B.1 that f_B for the T-junction does not follow the expected behavior after surpassing the gas superficial velocity of 0,259 m/s for the case of $U_{SL} = 0,531\ m/s$. Furthermore, from the figure 3.5, which reveals the bubble generation for both geometries, it can be observed how the bubble generation for the T-junction stops following the expected response after increasing U_{SL}. As for the discrepancies between the numerical simulations of both geometries, they alter between 3,5% to 22,4% for $U_{SL} = 0,318$ m/s, and between 28,7% and 58,4% for $U_{SL} = 0,531$ m/s. As for the fluctuations, they increase with the water superficial velocity ranging from 6,8 Hz to 56,9 Hz for the double T-junction, and from 3,7 Hz to 88,6 Hz for the T-junction. As a result, the errors and fluctuations are bigger for higher values of U_{SL}.

3.2. Bubble velocity

For the bubble velocity, the table B.2 contains the results for the different combinations of superficial velocities with their discrepancies and standard deviations. On the other

hand, by following the Drift-Flux model mentioned in the previous work of Arias [4], we have expressed the bubble velocity U_B as a function of the total air and water superficial velocities U_M as follows:

$$U_B = C_0\, U_M = C_0\, (U_{SL} + U_{SG}) \tag{3.2}$$

where C_0 is a coefficient that should always be greater than 1 in microgravity conditions since the U_B will be always bigger than the U_M. This is mainly owing to the bubble being faster than the water, which is always attached to the walls with a tending-to-zero velocity that decreases its overall velocity. We have obtained the coefficients $C_{0D} = 1,25$ and $C_{0T} = 1,18$ for the double T and T junctions respectively. Both values prove that the bubbles are faster than the mixture since they are not attached to the walls. From the figure 3.2 we can see the linear behavior of the bubble velocity, which increases with the increase of the total mixture velocity. In addition, qualitatively, the velocity results of the T and double T junctions are very similar, and quantitatively, the difference in U_B for both geometries ranges from 7,7% to 24,1% for $U_{SL} = 0,318$ m/s, and from 0,2% to 8,9% for $U_{SL} = 0,318$ m/s, thus the higher the liquid superficial velocity, the more similar the bubble velocities for the T and double T junctions. As for the standard deviation, it fluctuates between 0,006 m/s and 0,033 m/s for the double T-junction, and between 0,006 m/s and 0,043 m/s for the T-junction.

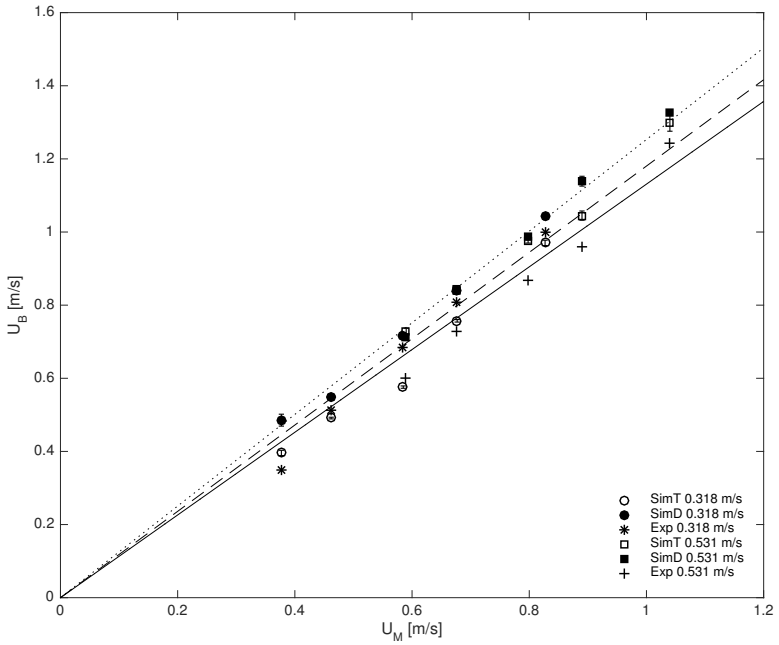

Figure 3.2: Bubble velocity as a function of the superficial gas velocity for two different superficial liquid velocities.

3.3. Bubble length

The bubble moves at its velocity U_B with the time required to form and detach a bubble, which is $1/f_B$. However, the bubble velocity can be replaced by the gas superficial velocity U_{SG} if the bubble fills the whole section of the pipe as explained in the page 114 of [6]. As a result, L_B can be defined as U_{SG}/f_B. In addition, we have used the pipe diameter to characterize the dimensionless length as follows:

$$\overline{L}_B = \frac{U_{SG}}{f_B\ \phi_C} \tag{3.3}$$

We have plotted the dimensionless bubble length as a function of $U_{SG}/f_B\ \phi_C$, which means that the results should follow a linear expression x=y, however, from the figure 3.3 we can see that the fitting lines of the T and double T junctions do not intersect with the origin x=y=0. Instead, $\overline{L}_B$ follows the following expression:

$$\overline{L}_B = C_1 + C_2\ \frac{U_{SG}}{f_B\ \phi_C} \tag{3.4}$$

where, C_1 and C_2 are the fitting constants. For the T-junction, $C_{1T} = 0,34$ and $C_{2T} = 1,37$. As for the double T-junction, $C_{1D} = 0,28$ and $C_{2D} = 1,53$.

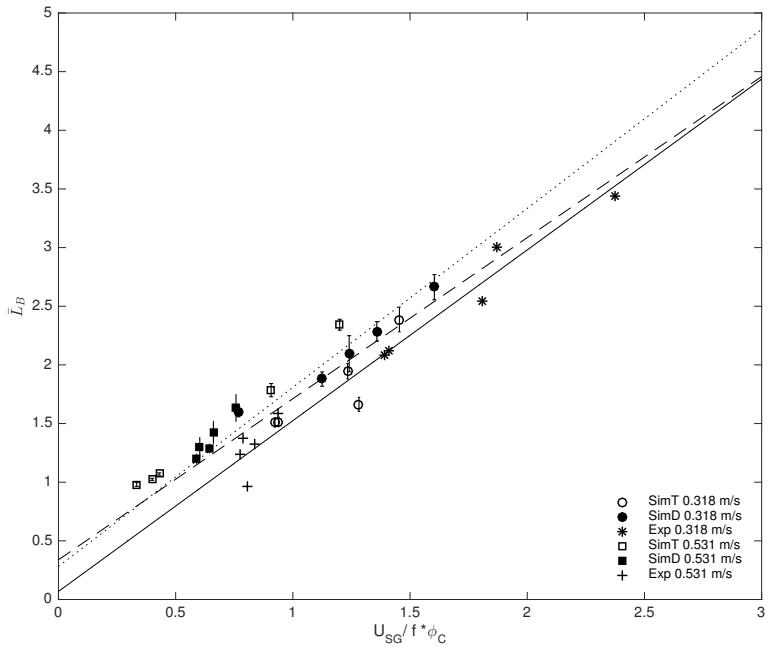

Figure 3.3: Bubble length as a function of the superficial gas velocity for two different superficial liquid velocities.

The linear tendency of the bubble length shows that it increases with the air superficial velocity. Also, the figure 3.5 illustrates how the length of the bubbles increases with U_{SG} and decreases with the increase of the liquid superficial velocity U_{SL}. Furthermore, the length gets larger gradually for the double T-junction. Nevertheless, in the case of

the T-junction, from one combination of $U_{SL} - U_{SG}$ to a next one, the increase can be gradual or abrupt as presented in the table B.3. It is related to the instability of the simulations for this geometry, which is reflected in the different bubble parameters. As for the errors between the T and double T junctions, they range from 5,7% to 33,1%. The standard deviation fluctuates between 0,075 m and 0,308m for the double T-junction, and between 0,075 m and 0,211 m for the T-junction.

3.4. Bubble volume

Lastly, we will discuss the bubble volume which is presented in the table B.4. The volume flow rate Q can be defined as the bubble generation frequency f_B times the bubble volume V_B, and also the gas superficial velocity U_{SG} times the cross-sectional area A of the pipe (from both expressions, the Q units are m^3/s). Consequently, the bubble volume is presented as:

$$V_B = \frac{U_{SG}\, A}{f_B} \tag{3.5}$$

Since we have plotted the dimensionless bubble volume, by dividing the previous equation by the cross-sectional area times the pipe diameter, we have expressed it as:

$$\overline{V}_B = \frac{U_{SG}}{f_B\, \phi_C} \tag{3.6}$$

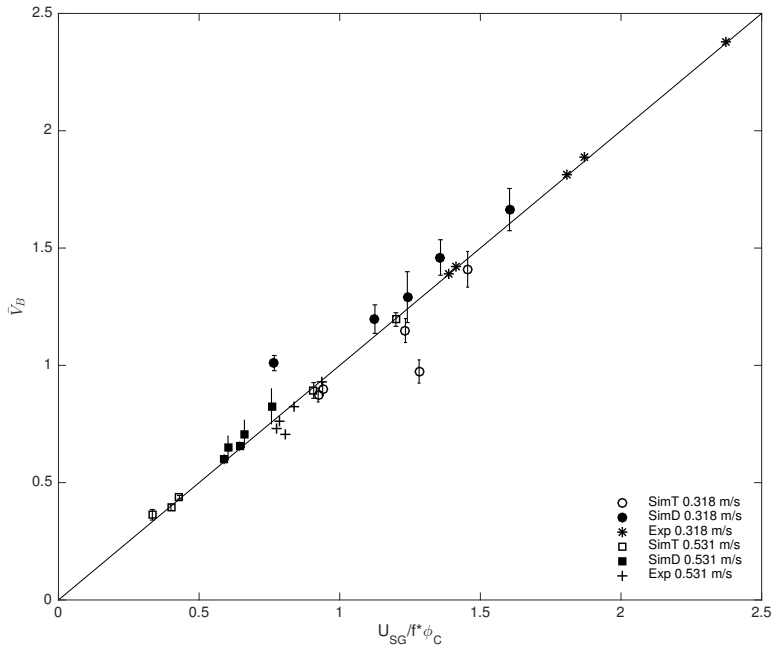

Figure 3.4: Bubble volume as a function of the superficial gas velocity for two different superficial liquid velocities.

In this case, as it can be seen in the figure 3.4, $\overline{V}_B$ follows a linear tendency. Most of the

values for both geometries are close to the x=y line. However, some elements deviate from this line. From the figure 3.5 we can see how that the bubble volume is often similar for both geometries, except for the combinations of $U_{SG} = 0,352$ m/s and $U_{SG} = 0,505$ m/s, where the bubbles are clearly bigger for the T-junction. One explanation could be that the bubble generation for the T-junction for higher liquid superficial velocities tends to be irregular. There could also be another explanation that is even simpler, and that is that these 2 simulations of Dalfó [10] are not credible (and should be discarded), for whatever reason (we have to improve the mesh, the time step, etc.).

The bubble volume for the double T-junctions increases stably with the increase of the air superficial velocity, but not the same for the T-junction. Furthermore, we could say that, unlike the double T-junction, the T-junction is more unstable and sensible to the increase of the liquid superficial velocity due to having only one water vertical inlet, which makes it asymmetric when it comes to cutting the gas and create a bubble. However, the volume results for the double T-junction in the case of $U_{SL} = 0,318$ m/s were all above the x=y line, which means bigger volume than in the T-junction and experiments cases. One logical explanation could be that, since half of the total U_{SL} is used in each vertical inlet, the water looses the impact of cutting the gas from only one side. As a result, it takes more time to finally cleave the flowing gas, therefore, the bubbles are larger yet stably generated. As for the errors, they range from 8,3% to 66,7%. The standard deviation varies from 0,027 m^3 to 0,17 m^3 for the double T-junction, and from 0,012 m^3 to 0,12 m^3 for the T-junction.

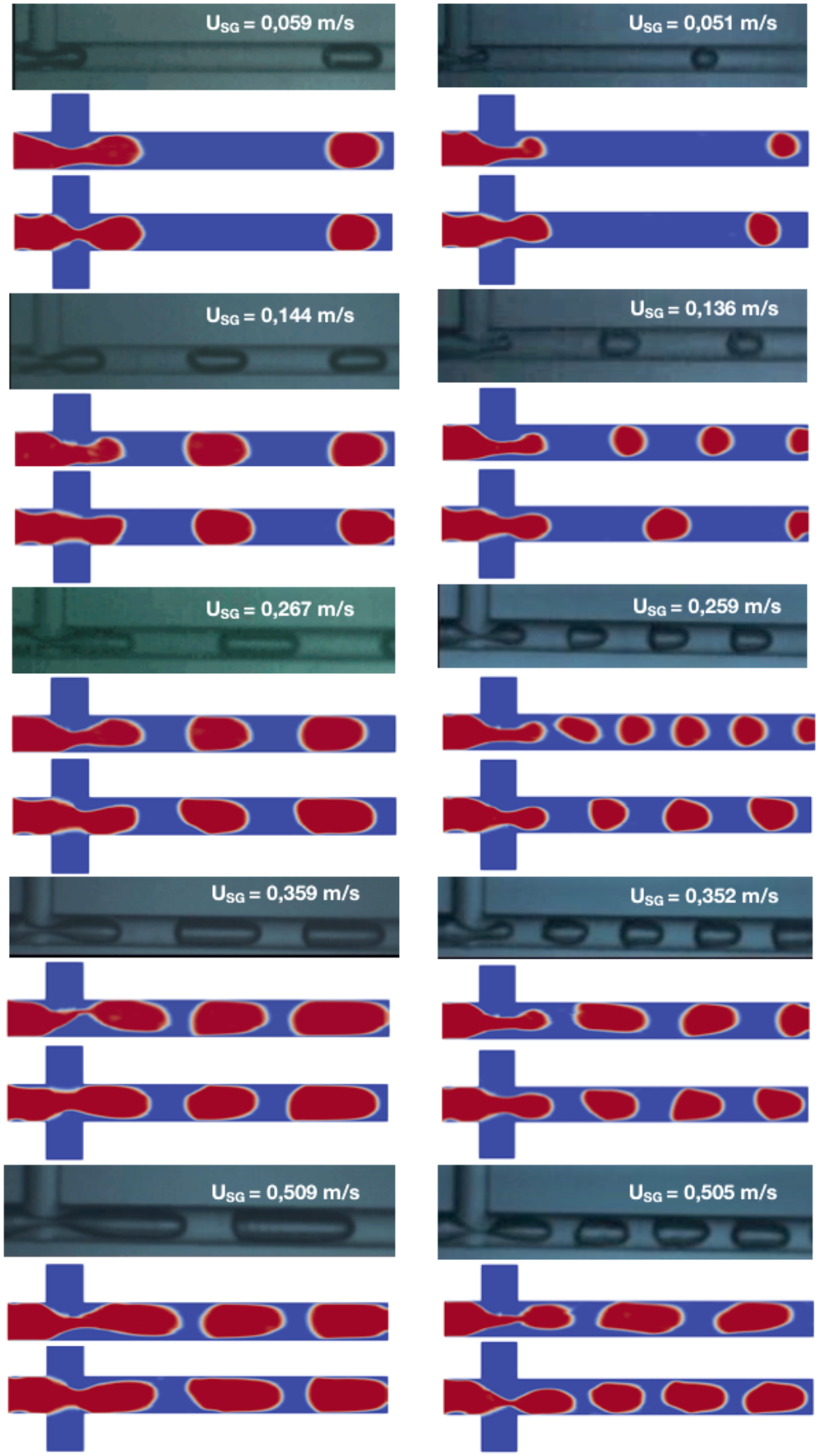

Figure 3.5: Bubble generation for the experiments, the T, and the double T junctions for the liquid superficial velocities 0,318 m/s (left) and 0,531 m/s (right)

CONCLUSIONS

In this project, we have studied the 3D numerical simulations of the bubble generation in a double T junction with an air-water mixture, and compared it with the numerical results of the T-junction from the project of Dalfó [10]. The study has been performed under microgravity conditions due to the current lack of knowledge. We have performed the simulations with the open-source program OpenFOAM, which has proved to be very efficient and flexible when it comes to the modification of different factors that affect the results such as the boundary conditions, physical properties, etc. during this work, the bubble generation frequency, velocity, length, and volume have been analyzed under different conditions for numerous combinations of air and water superficial velocities in order to discuss their great effects on the bubble behavior.

For the validations phase, different convergence tests were performed for the T-junction, to decide on a mesh of around 400.000 grids for both geometries, as a balance between the duration of the simulations, and the quality of the bubble generation. Furthermore, many tests were done to solve the critical issue of bubble attachment to the walls, such as the contact angle and the wettability condition. The problem was finally puzzled out with the wettability condition, and combined with the addition of a boundary layer in the mesh, more stability could be observed in the bubble generation process. However, a lot of future work is needed until we have simulations that faithfully reproduce the experiments. In addition, several tests were carried out with the double T-junction to see the effects of different contact angles on the bubbles. From all these tests for both geometries, it has been concluded that the different solutions were only effective for some combinations of superficial velocities. In addition, in the last part of the validations chapter it was concluded that, unlike for the T-junction, the bubble generation was easily stabilized with the double T-junction after adding the contact angle since there was no excessive fluctuations nor gas dispersion.

For the chapter of results, the contact angle condition was put aside since its effects were not consistent enough for both geometries. Subsequently, we have used the wettability condition with the new combinations of water and air superficial velocities. We have simulated the water superficial velocities 0,318 m/s and 0,531 m/s with five air superficial velocities each. One of the important conclusions is that the T-junction is more sensible to the change in the gas and liquid superficial velocities, therefore, the bubble generation of the T-junction for higher water superficial velocities tend to be irregular, and it is more reflected in the bubble frequency, length and volume. Furthermore, the bubbles are more stable and well detached from the walls for the double T-junction, due to the fact of having two vertical water inlets, which make it easier for the gas to be cut in order to create a well-shaped bubble not attached to the walls. One explanation is that the upper and lower sides of the gas are receiving the same quantity of water impact, unlike in the asymmetric T-junction in terms of the vertical inlet. However, the double T-junction has one inconvenient that was found after analyzing the bubble volume in the case of $U_{SL} = 0,318$ m/s. The volume results were all higher than the ones of the T-junction and experiments due to the reduced impact of the water after dividing its superficial velocity, which caused the flowing gas to need more time to be split, hence, the bubbles were larger yet stably generated.

Finally, This study has proved the complexity of the numerical simulation of multi-phase flows, which clearly stand out with their great potential. One clear example is the contact angle as one of the most promising boundary conditions that still needs a lot of research and evaluations due to the ambiguity of its effects when combined with other conditions. Also, changing the geometry configuration can greatly improve the bubble generation stability as seen with the double T-junction in some specific situations. Nevertheless, this study still needs more future investigations in order to not only keep improving the different bubble parameters but also to solve the different faced uncertainties.

BIBLIOGRAPHY

[1] McQuillen, J., Colin, C., Fabre, J. *Ground-based gas-liquid flow research in micro-gravity conditions*. (National Aeronautics and Space Administration, Lewis Research Center, and Institut de Mécanique des Fluides de Toulouse). https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20000004785.pdf (07-2018) 5

[2] McQuillen, J. *Two Phase Flow and Space-based Applications*. (Institute for Multifluid Science and Technology, Microgravity Fluid Physics Branch, NASA Glenn Research Center. 1999). 5

[3] Arias, S., De Montlaur, A. *Influence of Contact Angle Boundary Condition on CFD simulation of T-junction*. Micro-gravity science and technology 1–9. (2018) 6, 16, 21, 37

[4] Arias, S., De Montlaur, A. *Numerical Study and Experimental Comparison of Two-Phase Flow Generation in a T-Junction*. AIAA journal 55(5), 1565–1574. (2017) 6, 19, 21, 33, 37, 38, 39

[5] Free Software Foundation, Inc. *OpenFOAM Programmer's Guide*. (Edition 1.2. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. 2002). 10

[6] Arias, S. *An analysis of two-phase flows in conditions relevant to micro-gravity*. (Edicions UPC. Spain. 2011). 16, 40

[7] József Nagy. *Multiphase (VOF) Simulation Project*. Institute of Polymer Injection Molding and Process Automation, Johannes Kepler University Linz, Austria. (2017) 9

[8] Kroshilin A.E., Kroshilin V.E., Kohut, P. *Analysis of two-phase flow models with two momentum equations*. (All-Russian Research Institute for Nuclear Power Plants VNIIAES). 9

[9] Young, D., Munson, B., Okiishi, T. "Viscous flow in pipes".*A brief introduction to fluid mechanics*. (Wiley International Edition. United States of America. 2004). 9

[10] Dalfó, B. *CFD study of the bubble generation process in a T-junction with inverse flows*. 37, 38, 42, 45

[11] OpenFOAM installation. https://www.openfoam.com/download/install-binary-mac.php (11-2017) 51

[12] Beginner's Linux Cheat Sheets. http://chembytes.wdfiles.com/local--files/cheatsheets/linux4beginner.pdf (11-2017) 51

[13] CFD Online Forum. https://www.cfd-online.com (06-2018)

[14] CFD DIRECT OpenFOAM User Guide. https://cfd.direct/openfoam/user-guide/ (06-2018) 51

[15] OpenFOAM GITHUB guide. https://github.com/OpenCFD/ (06-2018) 51

[16] OpenFOAM fvSolution file. https://cfd.direct/openfoam/user-guide/fvsolution/ (06-2018) 53

[17] OpenFOAM fvSchemes file. https://cfd.direct/openfoam/user-guide/fvschemes/ (06-2018) 53

[18] Hernandez, I. *Comparison between 3D experimental and numerical simulations in the process of bubble generation for a gas-liquid combination in a T-junction mini-channel*. 2016 69

APPENDICES

APPENDIX A. OPENFOAM

Here we will explain in more detail the different steps of the OpenFOAM simulations for our project, and at the same time, we will describe the main modified files. More information can be found in the OpenFOAM user guide [14]. As for the installation of OpenFOAM, we have followed the steps on the website [11] (for Mac, Windows, and Linux). To learn some of the most important Linux commands that are used in the terminal for OpenFOAM, read the sheets [12].

A.1. Pre-prossesing

Before starting the simulations, we create the case with its corresponding, geometry, mesh and numerical conditions. OpenFOAM contains many examples that can be modified to match our study, which can be found in the GITHUB guide [15]. We have modified the following folders below:

The 0 folder:

The 0 folder and its data can be either, initial values and boundary conditions that the user must specify to define the problem; or, results written to the postProcessing file that will be described in the next section. We have defined the following files for our two-phase laminar flow case:

- alpha.air A.1: The dimensions vector is 0 because the air fraction is dimensionless. The internal field is 0 because we define the pipes to be full of water as an initial condition. Lastly we the boundary field by putting the boundary conditions related to the air and water fractions, depending on the part of the geometry.
- p_{rgh} A.2: Here we define the pressure boundary and initial conditions
- U A.3: Here we define the water and air superficial velocities.

The Constant folder:

The Constant folder that contains a full description of the case mesh and the physical properties. We will describe:

- g A.4: The gravity file where we define the vector to be (0 0 0) since our study is in microgravity conditions.
- boundary A.5: When we create our mesh with ICEM, we obtain a "name.msh" that, with the command "Fluent3DMeshToFoam" in the OpenFOAM terminal, we obtain the polyMesh folder with many files such as the boundary file, which is a list of patches, each of which is associated with a boundary condition. A patch is a list of face labels which clearly must contain only boundary faces and no internal faces. The names of the geometry parts are created automatically as they were initially defined in Solidworks, in ICEM, or manually by modifying the ".stl" file

of the geometry. These names in the boundary file can be changed to match the ones in the files of the 0 folder.

- transportProperties A.6: It specifies physical properties for the application concerned, in our case for the water and the air.

- turbulenceProperties A.8: In this file, we specify whether our case will be laminar or turbulent. Our case is laminar, thus we activate the "laminar" option of the "simulationType" keyword entry.

The Systems folder:

The Systems folder is used to set the parameters associated with the solution procedure itself. It contains different files, and we will describe the following ones that needed to be modified:

- controlDict A.12: This dictionary sets the input parameters that are essential for the creation of the database. The keyword entries in controlDict are listed down below:

  - startFrom: It controls the start time of the simulation (firstTime, startTime, or latestTime)
  - startTime: Since we start our simulations from the "startTime" option, which means any keyword entry we want, we start at 0 seconds.
  - stopAt: Controls the end time of the simulation (endTime, writeNow, nowWriteNow, or nextWrite)
  - endTime: End time for the simulation when stopAt endTime; is specified, in our case 2 seconds.
  - deltaT: It is the time step of our simulations.
  - writeControl: It controls the timing of writing an output to file (in the 0/processors folders). in this case, we use real clock time timings.
  - writeInterval: Every 900 seconds of real clock time.
  - MaxAlphaCo: The maximum value of the Courant number that our simulations can accept without aborting.
  - functions: Here we specify the location, writing time and different parameters related to the monitors where our bubble information will be detected. This information will be saved in the "faceSource.dat" file inside the postProcessing folder.

- setFieldsDict A.8: Here we define the volume of gas that we want initially. We have created a cylinder of 0,0005 meters length, and 0,0005 m of radius, to be placed in the inlet1 of the horizontal pipe.

- decomposeParDict A.9: This file is used for the decomposition of the mesh and the initial field data since we have simulated our cases with four cores in parallel. We used the "simple" keyword entry but there are the "hierarchical", "scotch", or "manual".

- fvSchemes A.10: IN this file we set the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation. More details in [17]
- fvSolution A.11: It controls the equation solvers, tolerances, and algorithms. More information in [16]

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha.air;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    WALLS1
    {
        type            fixedValue;
        value           uniform 0;
    }
    WALLS2
    {
        type            fixedValue;
        value           uniform 0;
    }
    WALLS3
    {
        type            fixedValue;
        value           uniform 0;
    }
    WALLS4
    {
        type            fixedValue;
        value           uniform 0;
    }
    INLET1
    {
        type            fixedValue;
        value           uniform 1;
    }
    INLET2
    {
        type            fixedValue;
        value           uniform 0;
    }
    INLET3
    {
        type            fixedValue;
        value           uniform 0;
    }
    OUTLET
    {
        type            zeroGradient;
    }
}
```

Figure A.1: The alpha.air file of the 0 folder

A.2. Processing

After creating the geometry, mesh, and setting all the files, we start the simulations in the terminal or our computer. The following commands are the steps to run our

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

 dimensions      [1 -1 -2 0 0 0 0];
 internalField   uniform 0;
 boundaryField
 {
    INLET1
    {
        type            zeroGradient; //air
    }
    INLET2
    {
        type            zeroGradient; //water
    }
    INLET3
    {
        type            zeroGradient; //water
    }
    OUTLET
    {
        type            fixedValue;
        value           uniform 0;
    }
    WALLS1
    {
        type            zeroGradient;
    }
    WALLS2
    {
        type            zeroGradient;
    }
    WALLS3
    {
        type            zeroGradient;
    }
    WALLS4
    {
        type            zeroGradient;
    }
  }
```

Figure A.2: The p_{rgh} file of the 0 folder

simulations in the case of parallel computations of 4 cores:

- **ssh user@IP**: We first connect with the cluster with interest.
- **transformPoints -scale '(0.001 0.001 0.001)'**: If we need to scale our geometry. It modifies the polyMesh folder inside the Constant folder. If we don't need to do it, we proceed to the next step.
- **cp 0/alpha.air.org 0/alpha.air**: With this command, we make a copy of the original alpha. air file that will be modified because of the parallel simulations.
- **setFields**: To set the cylinder of gas created in the air inlet.
- **decomposePar**: To decompose the simulation. With this command, the four processors that appear in the figure 1.1 are created.

- **nohup mpirun -np 4 interFoam -parallel >std.txt &**: This last command start the simulation with the interFoam solver. it saves the iterations in a "std.txt" file, where we can see the time step number of iterations, the Courant number, and different parameters while the simulations are running. We can also see the errors that cause the simulation abortion of we add an "err.txt" file.

If we run the simulations without the OpenFOAM parallelism, we only need to write directly **interFoam** in the terminal without the rest of commands. Also, one of the important commands to know before running a simulation is the **checkMesh** command that allows us to check the quality of the mesh, the number of cells, the skewness, the non-orthogonality, the patches, etc.

After starting the simulations, the processing phase starts. In this step, the postProcessing folders is created and it start ti store the simulation information specified in the controlDict file, which is the air fraction α in the monitors located at $x = 0,007\, m$ and $x = 0,008\, m$, as a function of the time-step. The figure A.13 shows the postProcessing file and its stored data. The figure A.14 shows the faceSourse.dat.

A.3. Post-Processing

In this step, we analyze the data obtained from the simulation. We can view the bubble generation, mesh, geometry, pressure, velocity, etc., with Paraview. We will explain the steps to open the results in Paraview by following the figure A.15, which shows the steps to open the data, and then the figure A.16, which shows how to set everything to finally visualize the bubble generation.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


 dimensions      [0 1 -1 0 0 0 0];
 internalField   uniform (0 0 0);
 boundaryField
 {
    INLET1
    {
                type            fixedValue; //air
                value           uniform (0.144 0 0);
    }
    INLET2
    {
                type            fixedValue; //water
                value           uniform (0 -0.159 0);
    }
    INLET3
    {
                type            fixedValue; //water
                value           uniform (0 0.159 0);
    }
    OUTLET
    {
                type            zeroGradient;
    }
    WALLS1
    {
                type            fixedValue;
                value           uniform (0 0 0);
    }
    WALLS2
    {
                type            fixedValue;
                value           uniform (0 0 0);
    }
    WALLS3
    {
                type            fixedValue;
                value           uniform (0 0 0);
    }
    WALLS4
    {
                type            fixedValue;
                value           uniform (0 0 0);
    }
 }
```

Figure A.3: The U file of the 0 folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


dimensions      [0 1 -2 0 0 0 0];
value           (0 0 0);  //no gravity considered
```

Figure A.4: The g (gravity) file of the Constant folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  3.0.1                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

8
(
    INLET1
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2117;
        startFace       1357026;
    }
    INLET2
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2117;
        startFace       1359143;
    }
    INLET3
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2117;
        startFace       1361260;
    }
    OUTLET
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2117;
        startFace       1363377;
    }
    WALLS1
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2204;
        startFace       1365494;
    }
    WALLS2
    {
        type            wall;
        inGroups        1(wall);
        nFaces          2204;
        startFace       1367698;
    }
    WALLS3
    {
        type            wall;
        inGroups        1(wall);
        nFaces          3045;
        startFace       1369902;
    }
    WALLS4
    {
        type            wall;
        inGroups        1(wall);
        nFaces          16965;
        startFace       1372947;
    }
)
```

Figure A.5: The boundary file of the System folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

 phases (air water);

 air
 {
    transportModel  Newtonian;
    nu              8.163e-06;
    rho             1.225;
 }

 water
 {
    transportModel  Newtonian;
    nu              0.000001;
    rho             1000;
 }

 sigma              0.072;
```

Figure A.6: The transportProperties file of the Constant folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;
```

Figure A.7: The turbulenceProperties file of the Constant folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha.air 0
);

regions
(
        cylinderToCell
    {
                p1 (-5 0 0); //first point of cylinder
                p2 (-4.9995 0 0); //last point of cylinder
                radius 0.0005;
        fieldValues
        (
            volScalarFieldValue alpha.air 1
        );
    }
);
```

Figure A.8: The setFieldsDict file of the System folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 4;  //numero de cores de l'ordinador

method          simple;

simpleCoeffs
{
    n               (2 2 1);
    delta           0.001;
}

hierarchicalCoeffs
{
    n               (1 1 1);
    delta           0.001;
    order           xyz;
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots           ( );
```

Figure A.9: The decomposeParDict file of the System folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes                      //first and second time derivatives
{
    default         Euler;
}

gradSchemes                     //gradient
{
    default         Gauss linear;
}

divSchemes                      //divergence
{
    div(rhoPhi,U)  Gauss linearUpwind grad(U);
    div(phi,alpha)  Gauss vanLeer;
    div(phirb,alpha) Gauss linear;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes                //Laplacion
{
    default         Gauss linear corrected;
}

interpolationSchemes            //cell to face interpolations of values
{
    default         linear;
}

snGradSchemes                   //component of gradient normal to a cell face
{
    default         corrected;
}
fluxRequired
{
    default         no;
    p_rgh;
    pcorr;
    alpha.air;
}
```

Figure A.10: The fvSchemes file of the System folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "alpha.air.*"
    {
        nAlphaCorr      2;
        nAlphaSubCycles 1;
        cAlpha          1;

        MULESCorr       yes;
        nLimiterIter    5;

        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-8;
       relTol          0;
    }
    "pcorr.*"
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-5;
        relTol          0;
    }
    p_rgh
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-07;
        relTol          0.05; //possar a 0 si ens falles en la simulació en paral·lel
    }
    p_rghFinal
    {
        $p_rgh;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }
}
PIMPLE
{
    momentumPredictor   no;
    nOuterCorrectors    1;  //mes gran a 1 si ens falles en la simulació en paral·lel
    nCorrectors         3;
    nNonOrthogonalCorrectors 0;
        cAlpha          1;
        nAlphaCorr      2;
        nAlphaSubCycles 5;
}
relaxationFactors
{
        "U.*"           1;
                "pcorr.*"               1;

}
```

Figure A.11: The fvSolution file of the System folder

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  plus                                  |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     interFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         2;
deltaT          0.000005;
writeControl    clockTime;
writeInterval   900;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable yes;
adjustTimeStep  no;
maxCo           2;
maxAlphaCo      1;
functions
{
        surface8
    {
        type            faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl    timeStep;// outputTime;
        log             true;
        valueOutput     false;
        source          sampledSurface;  // Type of face source: faceZone, patch
        sourceName      surface8;
        operation       average;  //areaAverage
        sampledSurfaceDict
        {
                type cuttingPlane;
                planeType          pointAndNormal;
                pointAndNormalDict
                {
                        basePoint       (0.003 0 0);
                        normalVector    (1 0 0);
                }
                source cells; // sample cells or boundaryFaces
                interpolate false;
        }
        fields
        (
                        alpha.air
        );
        surfaceFormat dx;
    }
            surface7
           {
                type            faceSource;
                functionObjectLibs ("libfieldFunctionObjects.so");
                enabled         true;
                outputControl    timeStep;// outputTime;
                log             true;
                valueOutput     false;
                source          sampledSurface;  // Type of face source: faceZone, patch
                sourceName      surface7;
                operation       average;  //areaAverage
                sampledSurfaceDict
                {
                        type cuttingPlane;
                        planeType          pointAndNormal;
                        pointAndNormalDict
                        {
                                                        basePoint       (0.002 0 0);
                                normalVector    (1 0 0);
                        }
                        source cells; // sample cells or boundaryFaces
                        interpolate false;
                }
                fields
                (
                                alpha.air
                );
                surfaceFormat dx;
           }
      }
```

Figure A.12: The controlDict file of the System folder

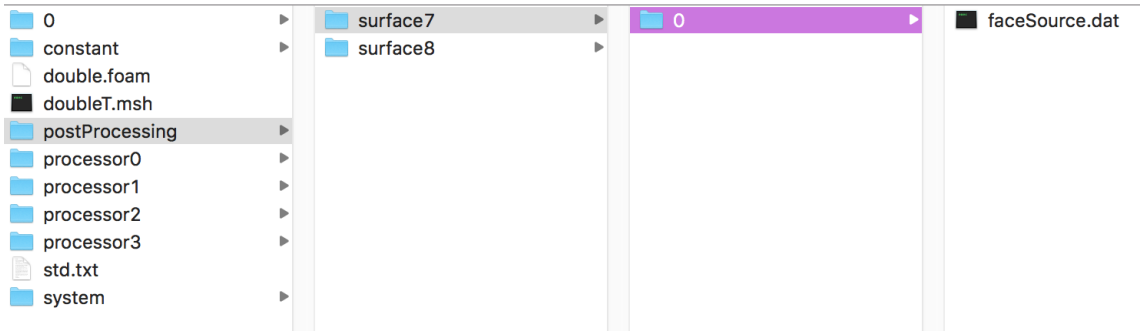

Figure A.13: The postProcessing folder

```
#   Source : sampledSurface surface7
#   Faces  : 8974
#   Area   : 7.904858e-07
#        Time   average(alpha.air)
5.000000e-06    6.726473e-198
1.000000e-05    1.635174e-196
1.500000e-05    2.759046e-195
2.000000e-05    3.913146e-194
2.500000e-05    4.946973e-193
3.000000e-05    5.732340e-192
3.500000e-05    6.190609e-191
4.000000e-05    6.298224e-190
4.500000e-05    6.087977e-189
5.000000e-05    5.638948e-188
5.500000e-05    5.054087e-187
6.000000e-05    4.432660e-186
6.500000e-05    3.847590e-185
7.000000e-05    3.335467e-184
7.500000e-05    2.899898e-183
8.000000e-05    2.524182e-182
8.500000e-05    2.185886e-181
9.000000e-05    1.867745e-180
9.500000e-05    1.562438e-179
1.000000e-04    1.271879e-178
1.050000e-04    1.003431e-177
1.100000e-04    7.654432e-177
1.150000e-04    5.639894e-176
1.200000e-04    4.013182e-175
1.250000e-04    2.758917e-174
1.300000e-04    1.833752e-173
1.350000e-04    1.179503e-172
1.400000e-04    7.349561e-172
1.450000e-04    4.441104e-171
1.500000e-04    2.605262e-170
1.550000e-04    1.485245e-169
1.600000e-04    8.237028e-169
1.650000e-04    4.448260e-168
1.700000e-04    2.341309e-167
1.750000e-04    1.202151e-166
1.800000e-04    6.026329e-166
1.850000e-04    2.951784e-165
1.900000e-04    1.413773e-164
1.950000e-04    6.625922e-164
2.000000e-04    3.040718e-163
2.050000e-04    1.367243e-162
2.100000e-04    6.027213e-162
2.150000e-04    2.606368e-161
2.200000e-04    1.106210e-160
```

Figure A.14: The faceSource file of the postProcessing folder

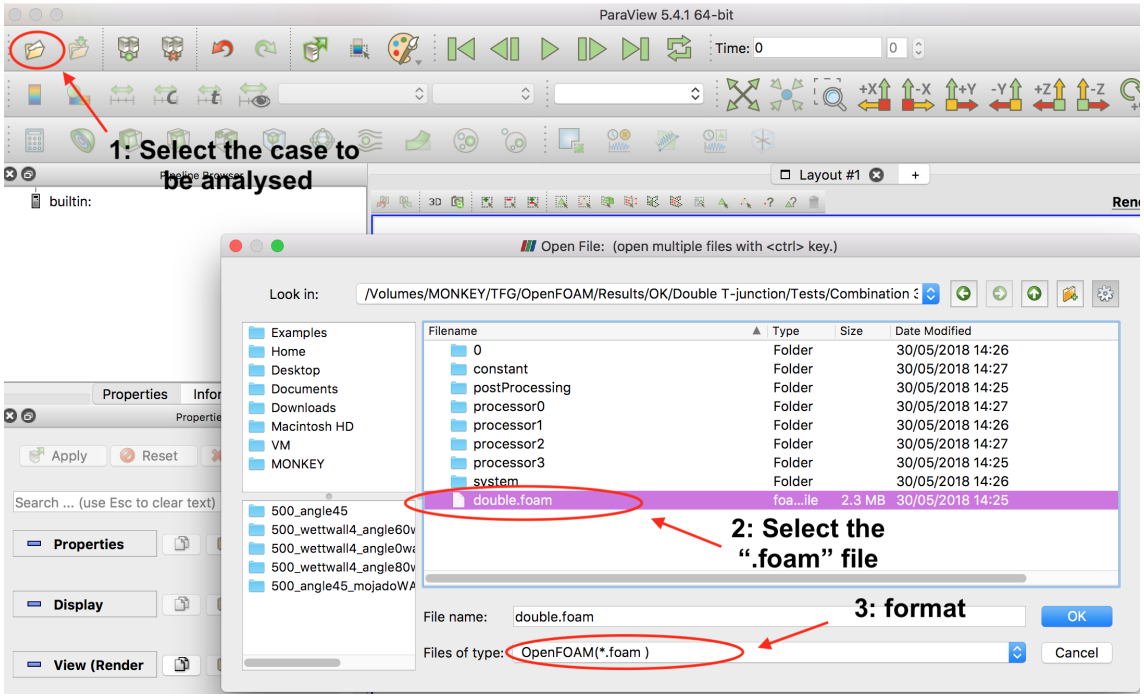

Figure A.15: The steps of the case selection in Paraview

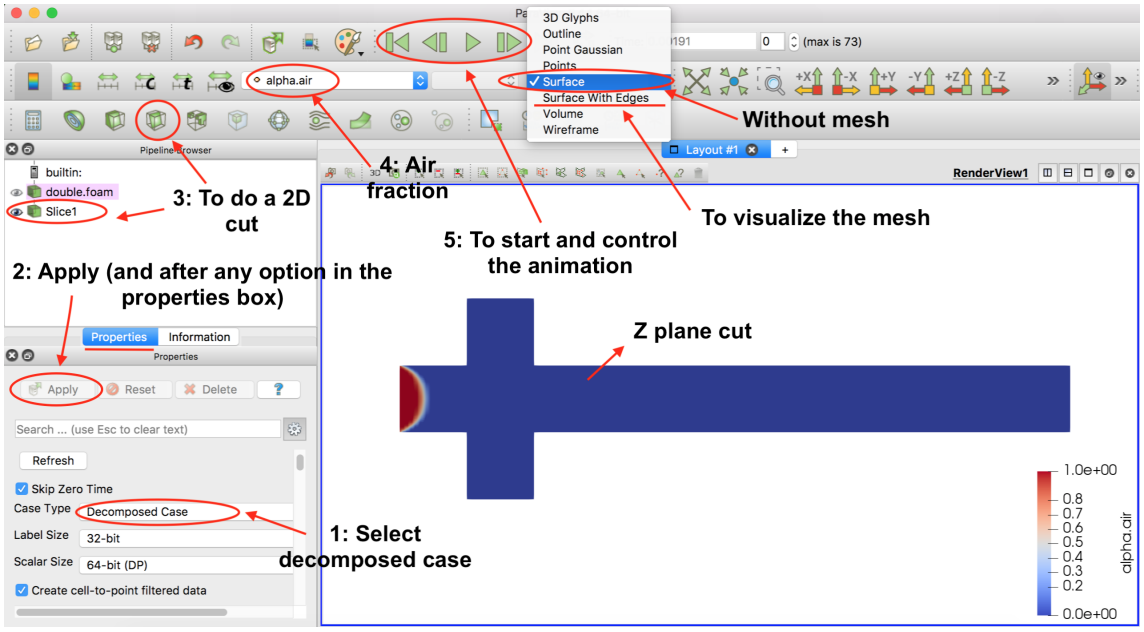

Figure A.16: The steps of the case preparation in Paraview

APPENDIX B. TABLES

$U_{SL}-U_{SG}$ m/s	f_{BE} Hz	f_{BD} Hz	$\sigma_{f_{BD}}$ Hz	f_{BT} Hz	$\sigma_{f_{BT}}$ Hz	ϵ_{fBT} %
0,318−0,059	41,81	76,89	7,723	62,81	3,669	22,4
0,318−0,144	103,72	127,98	14,683	155,94	8,061	17,9
0,318−0,267	147,78	215,27	6,799	208,00	12,000	3,5
0,318−0,359	191,93	264,30	16,680	291,03	8,214	9,2
0,318−0,509	214,42	317.39	15,632	349,89	27,305	9,3
0,531−0,051	73,28	100,16	10,291	176,73	88,554	43,3
0,531−0,136	185,83	223,61	42,474	360,09	33,954	37,9
0,531−0,259	339,83	442,87	32,432	621,09	14,973	28,7
0,531−0,352	427,63	543,03	38,964	395,66	17,767	37,2
0,531−0,505	545,17	672,10	56,888	424,26	10,951	58,4

Table B.1: The bubble generation frequency, its standard deviation, and error for the different combinations of velocities, for the T and double T junctions

$U_{SL}-U_{SG}$ m/s	U_{BE} m/s	U_{BD} m/s	$\sigma_{U_{BD}}$ m/s	U_{BT} m/s	$\sigma_{U_{BT}}$ m/s	ϵ_{UBT} %
0,318−0,059	0,350	0,485	0,033	0,396	0,015	22,5
0,318−0,144	0,514	0,549	0,012	0,491	0,005	11,8
0,318−0,267	0,684	0,715	0,015	0,576	0,006	24,1
0,318−0,359	0,807	0,839	0,013	0,757	0,007	10,8
0,318−0,509	1,000	1,045	0,006	0,970	0,015	7,7
0,531−0,051	0,602	0,712	0,016	0,727	0,020	2,1
0,531−0,136	0,730	0,841	0,024	0,843	0,014	0,2
0,531−0,259	0,869	0,986	0,016	0,977	0,014	0,9
0,531−0,352	0,960	1,139	0,027	1,045	0,025	8,9
0,531−0,505	1,241	1,328	0,014	1,297	0,043	2,4

Table B.2: The bubble velocity, its standard deviation, and error for the different combinations of velocities, for the T and double T junctions

$U_{SL}-U_{SG}$ m/s	L_{BE} m	L_{BD} m	$\sigma_{L_{BD}}$ m	L_{BT} m	$\sigma_{L_{BT}}$ m	ϵ_{LBT} %
0,318−0,059	2,120	1,601	0,087	1,514	0,079	5,7
0,318−0,144	2,080	1,879	0,122	1,508	0,075	24,6
0,318−0,267	2,538	2,095	0,308	1,664	0,119	25,9
0,318−0,359	3,000	2,285	0,166	1,946	0,132	17,4
0,318−0,509	3,444	2,663	0,215	2,388	0,211	11,5
0,531−0,051	0,961	1,307	0,141	0,982	0,034	33,1
0,531−0,136	1,240	1,288	0,075	1,025	0,018	25,6
0,531−0,259	1,370	1,300	0,159	1,070	0,019	21,5
0,531−0,352	1,321	1,425	0,188	1,785	0,113	20,2
0,531−0,505	1,592	1,633	0,227	2,342	0,093	30,3

Table B.3: The bubble length, its standard deviation, and error for the different combinations of velocities, for the T and double T junctions

$U_{SL}-U_{SG}$ m/s	$V_{BE}10^{-9}$ m^3	$V_{BD}10^{-9}$ m^3	$\sigma_{V_{BD}}$ m^3	$V_{BT}10^{-9}$ m^3	$\sigma_{V_{BT}}10^{-9}$ m^3	ϵ_{VBT} %
0,318−0,059	1,118	0,793	0,051	0,707	0,043	12,2
0,318−0,144	1,093	0,940	0,095	0,686	0,047	37,0
0,318−0,267	1,422	1,014	0,170	0,765	0,078	32,5
0,318−0,359	1,483	1,147	0,119	0,902	0,080	27,2
0,318−0,509	1,867	1,199	0,079	1,107	0,120	8,3
0,531−0,051	0,552	0,471	0,030	0,285	0,035	65,3
0,531−0,136	0,575	0,515	0,027	0,309	0,019	66,7
0,531−0,259	0,599	0,509	0,079	0,344	0,012	47,9
0,531−0,352	0,648	0,556	0,091	0,701	0,052	20,7
0,531−0,505	0,729	0,648	0,117	0,939	0,046	30,9

Table B.4: The bubble volume, its standard deviation, and error for the different combinations of velocities, for the T and double T junctions

APPENDIX C. MATLAB CODE FOR VALIDATIONS

This code is from the previous project of Hernàndez [18].

```
% monitor x=8mm:
clear all
load surf_mon_8.dat
ref=surf_mon_8;
% 1mm area
area=pi*((1e-3)/2)^2;
% air fraction
airref = ref(:,2);
% water fraction
wref = 1-airref;
% Time
tref=ref(:,1);
% reading of initial and final points of bubble
i=1;
k=1;
init=true;
valor=0.0015;%0.0015;
while(i<=length(airref))
    done=false;
    if(airref(i)>=valor&&init==true)
        points(k)=i;
        k=k+1;
        init=false;
        done=true;
    end
    if(airref(i)<=valor&&init==false&&done==false)
        points(k)=i;
        k=k+1;
        init=true;
    end
    i=i+1;
end
% odd points elimination
if(rem(length(points),2)==1)
    points(k-1)=[];
end
% inicial and final points separation
i=1;
k=1;
while(i<=length(points))
    ipoints(k)=points(i);
```

```
42      i=i+1;
43      fpoints(k)=points(i);
44      i=i+1;
45      k=k+1;
46  end
47  % bubble area
48  i=1;
49  while(i<=length(ipoints))
50      a(i)=trapz(tref(ipoints(i):fpoints(i)),airref(ipoints(i)
            :fpoints(i)));
51      av(i)=a(i)*area;
52      i=i+1;
53  end
54  % delete small bubbles (gas dispersion)
55  i=1;
56  while(i<=length(av))
57      if(av(i)<=10e-11)
58          fpoints(i)=[];
59          ipoints(i)=[];
60          av(i)=[];
61          a(i)=[];
62          i=i-1;
63      end
64      i=i+1;
65  end
66  % Frequency
67  i=1;
68  while(i<=length(ipoints)-1)
69      pii(i)=tref(ipoints(i+1))-tref(ipoints(i));
70      fi(i)=1/pii(i);
71      i=i+1;
72  end
73  % monitor x=7mm:
74  load surf_mon_7.dat
75  ref7=surf_mon_7;
76  % air fraction
77  airref7=ref7(:,2);
78  % water fraction
79  wref7 = 1-airref7;
80  % Temps:
81  tref7=ref7(:,1);
82  % reading of inicial and final bubble points
83  i=1;
84  k=1;
85  init=true;
86  valor=0.0015;
87  while(i<=length(airref7))
88      done=false;
```

```
89      if(airref7(i)>=valor&&init==true)
90          points7(k)=i;
91          k=k+1;
92          init=false;
93          done=true;
94      end
95      if(airref7(i)<=valor&&init==false&&done==false)
96          points7(k)=i;
97          k=k+1;
98          init=true;
99      end
100     i=i+1;
101 end
102 % odd points elimination
103 if(rem(length(points7),2)==1)
104     points7(k-1)=[];
105 end
106 % initial and final points separation
107 i=1;
108 k=1;
109 while(i<=length(points7))
110     ipoints7(k)=points7(i);
111     i=i+1;
112     fpoints7(k)=points7(i);
113     i=i+1;
114     k=k+1;
115 end
116 % surface
117 i=1;
118 while(i<=length(ipoints7))
119     a7(i)=trapz(tref7(ipoints7(i):fpoints7(i)),airref7(
          ipoints7(i):fpoints7(i)));
120     av7(i)=a7(i)*area;
121     i=i+1;
122 end
123 % delete small bubbles (gas dispersion)
124 i=1;
125 while(i<=length(av7))
126     if(av7(i)<=10e-11)
127         fpoints7(i)=[];
128         ipoints7(i)=[];
129         av7(i)=[];
130         a7(i)=[];
131         i=i-1;
132     end
133     i=i+1;
134 end
135 % bubble velocity and volume
```

```
i=1;
while(i<=length(ipoints)) %ipoints
    tbetween(i)=tref(ipoints(i))-tref7(ipoints7(i));
    Ug(i)=1e-3/tbetween(i);
    volume(i)=av(i)*Ug(i);
    i=i+1;
end
% bubble length
i=1;
while(i<=length(ipoints)) %ipoints
    tbubbles(i)=tref(fpoints(i))-tref(ipoints(i));
    L(i)=Ug(i)*tbubbles(i);
    i=i+1;
end
longitude=mean(L);
velocity=mean(Ug);
frequency=mean(fi);
%Vector to cut until xx bubbles
i=1;
while(i<=70)
    fi2(i)=fi(i);
    U2(i)=Ug(i);
    L2(i)=L(i);
    V2(i)=volume(i);
    i=i+1;
end
%Vector with last 5 or 6 bubbles
k=1;
for i=1:length(fi2)
    if(i>=65)
        finova(k)=fi2(i);
        Ugnova(k)=U2(i);
        Lnova(k)=L2(i)*1000;
        Vnova(k)=V2(i);
        k=k+1;
    end
    i=i+1;
end
%mean (5 bubbles for chap 2 and 6 bubbles for chap 3)
f_B=mean(finova)
L_B=mean(Lnova)
U_B=mean(Ugnova)
V_B=mean(Vnova)
%standard deviation
for i=1:length(finova)
    f(i)=(finova(i)-f_B)^2;
    l(i)=(Lnova(i)-L_B)^2;
    u(i)=(Ugnova(i)-U_B)^2;
```

```
184         v(i)=(Vnova(i)-V_B)^2;
185 end
186 df=sqrt(sum(f)/5)
187 dl=sqrt(sum(l)/5)
188 du=sqrt(sum(u)/5)
189 dv=sqrt(sum(v)/5)
190 %matrixed of bubble parameters and deviations
191 comb9d=[f_B,L_B,U_B,V_B];
192 dev9d=[df,dl,du,dv];
193 save('comb9d.mat','comb9d');
194 save('dev9d.mat','dev9d');
195 %PLOTS
196 figure(2)
197 subplot(4,1,1)
198 plot(tref,airref,'black')
199 xlabel('time [s]')
200 ylabel('fraction of air')
201 title('x=0.008m')
202 subplot(4,1,2)
203 plot(fi2)
204 xlabel('bubble')
205 ylabel('frequency [Hz]')
206 subplot(4,1,3)
207 plot(L2,'green')
208 xlabel('bubble')
209 ylabel('longitude [m]')
210 subplot(4,1,4)
211 plot(U2,'-r')
212 xlabel('bubble')
213 ylabel('velocity [m/s]')
214 %for alpha diagram
215 figure(3)
216 plot(tref, airref7,'-')
217 hold on;
218 plot(tref,airref,'--')
219 xlabel('Time [s]')
220 ylabel('Air fraction')
221 xlim([0.02,0.031])
222 ylim([0,1])
223 legend('x=0,007 m', 'x=0,008 m')
224 legend('boxoff')
```

APPENDIX D. MATLAB PLOTS CODE FOR VALIDATIONS

```
1
2  % Plots
3  figure(1)
4  subplot(3,1,1)
5  plot(fi21,':k','linewidth',1)
6  hold on
7  plot(fi22,'--k')
8  hold on
9  plot(fi23,'-k')
10 xlabel('Bubbles','fontweight','bold','FontSize',12)
11 ylabel('Frequency [Hz]','fontweight','bold','FontSize',12)
12 %xlim([1 16])
13
14 subplot(3,1,2)
15 plot(L21,':k','linewidth',1)
16 hold on
17 plot(L22,'--k')
18 hold on
19 plot(L23,'-k')
20 xlabel('Bubbles','fontweight','bold','FontSize',12)
21 ylabel('Length [m]','fontweight','bold','FontSize',12)
22 xlim([1 16])
23 %ylim([0 0.01])
24
25 subplot(3,1,3)
26 plot(U21,':k','linewidth',1)
27 hold on
28 plot(U22,'--k')
29 hold on
30 plot(U23,'-k')
31 xlabel('Bubbles','fontweight','bold','FontSize',12)
32 ylabel('Velocity [m/s]','fontweight','bold','FontSize',12)
33 %ylim([0 0.5])
34 %xlim([1 16])
35
36 legend('220.000 cells', '400.000 cells', '600.000 cells')
```

APPENDIX E. MATLAB PLOTS CODE FOR RESULTS

```
clear all;
close all;
%GRUP1
load('comb1.mat')
load('comb2.mat')
load('comb3.mat')
load('comb4.mat')
load('comb5.mat')
load('dev1.mat')
load('dev2.mat')
load('dev3.mat')
load('dev4.mat')
load('dev5.mat')
load('comb1d.mat')
load('comb2d.mat')
load('comb3d.mat')
load('comb4d.mat')
load('comb5d.mat')
load('dev1d.mat')
load('dev2d.mat')
load('dev3d.mat')
load('dev4d.mat')
load('dev5d.mat')
%%GRUP2
load('comb6.mat')
load('comb7.mat')
load('comb8.mat')
load('comb9.mat')
load('comb10.mat')
load('dev6.mat')
load('dev7.mat')
load('dev8.mat')
load('dev9.mat')
load('dev10.mat')
load('comb6d.mat')
load('comb7d.mat')
load('comb8d.mat')
load('comb9d.mat')
load('comb10d.mat')
load('dev6d.mat')
load('dev7d.mat')
load('dev8d.mat')
load('dev9d.mat')
```

```
load('dev10d.mat')

%% Parametres
Usg1=[0,0.059,0.144, 0.267, 0.359,0.509];
Usl1=0.318*ones(1,6); Usl(1)=0;

Usg2=[0,0.059,0.144, 0.267, 0.359,0.509];
Usl2=0.531*ones(1,6); Usl(1)=0;
diametre=0.001;

%EXPERIMENTALS
fE1=[0,41.806,103.721,147.778,191.930,214.425];
lE1=[0,2.120,2.080,2.538,3.000,3.444]/1000;
uE1=[0,0.350,0.514,0.684,0.807,1.000];
vE1=[0,1.118,1.093,1.422,1.483,1.867]*10^-9;

fE2=[0,73.282,185.832,339.827,427.632,545.167];
lE2=[0,0.961,1.240,1.370,1.321,1.592]/1000;
uE2=[0,0.602,0.730,0.869,0.960,1.241];
vE2=[0,0.552,0.575,0.599,0.648,0.729]*10^-9;

%SIMULACIONS T
f1=[0,comb1(1),comb2(1),comb3(1),comb4(1),comb5(1)];
l1=[0,comb1(2),comb2(2),comb3(2),comb4(2),comb5(2)]/1000;
u1=[0,comb1(3),comb2(3),comb3(3),comb4(3),comb5(3)];
v1=[0,comb1(4),comb2(4),comb3(4),comb4(4),comb5(4)];
devf1=[dev1(1),dev2(1),dev3(1),dev4(1),dev5(1)]/2;
devl1=[dev1(2),dev2(2),dev3(2),dev4(2),dev5(2)]/(2*1000);
devu1=[dev1(3),dev2(3),dev3(3),dev4(3),dev5(3)]/2;
devv1=[dev1(4),dev2(4),dev3(4),dev4(4),dev5(4)]/2;

f2=[0,comb6(1),comb7(1),comb8(1),comb9(1),comb10(1)];
l2=[0,comb6(2),comb7(2),comb8(2),comb9(2),comb10(2)]/1000;
u2=[0,comb6(3),comb7(3),comb8(3),comb9(3),comb10(3)];
v2=[0,comb6(4),comb7(4),comb8(4),comb9(4),comb10(4)];
devf2=[dev6(1),dev7(1),dev8(1),dev9(1),dev10(1)]/2;
devl2=[dev6(2),dev7(2),dev8(2),dev9(2),dev10(2)]/(2*1000);
devu2=[dev6(3),dev7(3),dev8(3),dev9(3),dev10(3)]/2;
devv2=[dev6(4),dev7(4),dev8(4),dev9(4),dev10(4)]/2;

%SIMULACIONS D
f1d=[0,comb1d(1),comb2d(1),comb3d(1),comb4d(1),comb5d(1)];
l1d=[0,comb1d(2),comb2d(2),comb3d(2),comb4d(2),comb5d(2)
    ]/1000;
u1d=[0,comb1d(3),comb2d(3),comb3d(3),comb4d(3),comb5d(3)];
v1d=[0,comb1d(4),comb2d(4),comb3d(4),comb4d(4),comb5d(4)];
devf1d=[dev1d(1),dev2d(1),dev3d(1),dev4d(1),dev5d(1)]/2;
devl1d=[dev1d(2),dev2d(2),dev3d(2),dev4d(2),dev5d(2)
```

```
    ]/(2*1000);
devu1d=[dev1d(3),dev2d(3),dev3d(3),dev4d(3),dev5d(3)]/2;
devv1d=[dev1d(4),dev2d(4),dev3d(4),dev4d(4),dev5d(4)]/2;

f2d=[0,comb6d(1),comb7d(1),comb8d(1),comb9d(1),comb10d(1)];
l2d=[0,comb6d(2),comb7d(2),comb8d(2),comb9d(2),comb10d(2)
    ]/1000;
u2d=[0,comb6d(3),comb7d(3),comb8d(3),comb9d(3),comb10d(3)];
v2d=[0,comb6d(4),comb7d(4),comb8d(4),comb9d(4),comb10d(4)];
devf2d=[dev6d(1),dev7d(1),dev8d(1),dev9d(1),dev10d(1)]/2;
devl2d=[dev6d(2),dev7d(2),dev8d(2),dev9d(2),dev10d(2)
    ]/(2*1000);
devu2d=[dev6d(3),dev7d(3),dev8d(3),dev9d(3),dev10d(3)]/2;
devv2d=[dev6d(4),dev7d(4),dev8d(4),dev9d(4),dev10d(4)]/2;

%% Plot Frequency
exponential1 = fittype('fsat*(1-exp(-ao*Usg1/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'f1'},'
    independent',{'Usg1'});
exponential1d = fittype('fsat*(1-exp(-ao*Usg1/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'f1d'},'
    independent',{'Usg1'});
exponentialE1 = fittype('fsat*(1-exp(-ao*Usg1/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'fE1'},'
    independent',{'Usg1'});
f_eq1=fit(Usg1',f1',exponential1)
f_eq1d=fit(Usg1',f1d',exponential1d)
fE_eq1=fit(Usg1',fE1',exponentialE1);

exponential2 = fittype('fsat*(1-exp(-ao*Usg2/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'f2'},'
    independent',{'Usg2'});
exponential2d = fittype('fsat*(1-exp(-ao*Usg2/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'f2d'},'
    independent',{'Usg2'});
exponentialE2 = fittype('fsat*(1-exp(-ao*Usg2/fsat))','
    coefficients',{'fsat','ao'},'dependent',{'fE2'},'
    independent',{'Usg2'});
f_eq2=fit(Usg2',f2',exponential2)
f_eq2d=fit(Usg2',f2d',exponential2d)
fE_eq2=fit(Usg2',fE2',exponentialE2);

figure (1)
%axis([0 0.55 0 600])
xlim([0,0.7])
ylim([0,700])
hold on;
plot(f_eq1,'--k')
```

```
124 hold on;
125 plot(f_eq1d,':k')
126 hold on;
127 plot(fE_eq1,'-k')
128 hold on;
129 plot(f_eq2,'--k')
130 hold on;
131 plot(f_eq2d,':k')
132 hold on;
133 plot(fE_eq2,'-k')
134 hold on;
135 h(1)=errorbar(Usg1(2:6),f1(2:6),devf1,'o','MarkerEdgeColor',
    'k','Color','k','LineWidth',0.75);
136 hold on;
137 h(2)=errorbar(Usg1(2:6),f1d(2:6),devf1d,'o','MarkerEdgeColor
    ','k','MarkerFaceColor','k','Color','k','LineWidth',0.75)
    ;
138 hold on;
139 h(3)=plot(Usg1(2:6),fE1(2:6),'*','MarkerEdgeColor','k','
    Color','k');
140 hold on;
141 h(4)=errorbar(Usg2(2:6),f2(2:6),devf2,'s','MarkerEdgeColor',
    'k','Color','k','LineWidth',0.75);
142 hold on;
143 h(5)=errorbar(Usg2(2:6),f2d(2:6),devf2d,'s','MarkerEdgeColor
    ','k','MarkerFaceColor','k','Color','k','LineWidth',0.75)
    ;
144 hold on;
145 h(6)=plot(Usg2(2:6),fE2(2:6),'+','MarkerEdgeColor','k','
    Color','k');
146 hold on;
147 xlabel('U_{SG} [m/s]')
148 ylabel('f_{B} [1/s]')
149 legend(h,'SimT 0.318 m/s','SimD 0.318 m/s','Exp 0.318 m/s','
    SimT 0.531 m/s','SimD 0.531 m/s','Exp 0.531 m/s','
    Location','southeast')
150 legend('boxoff')
151
152 %% Plot volume
153 for i=2:length(Usg1)
154     Vb1(i)=Usg1(i)/(f1(i)*diametre);
155     Vb2(i)=Usg2(i)/(f2(i)*diametre);
156     Vb1d(i)=Usg1(i)/(f1d(i)*diametre);
157     Vb2d(i)=Usg2(i)/(f2d(i)*diametre);
158     VbE1(i)=Usg1(i)/(fE1(i)*diametre);
159     VbE2(i)=Usg2(i)/(fE2(i)*diametre);
160 end
161 v_adim1=v1./(pi/4*(diametre)^3);
```

```
devv_adim1=devv1./(pi/4*(diametre)^3);
v_adim1d=v1d./(pi/4*(diametre)^3);
devv_adim1d=devv1d./(pi/4*(diametre)^3);
v_adimE1=vE1./(pi/4*(diametre)^3);
funct2=@(x) x;

v_adim2=v2./(pi/4*(diametre)^3);
devv_adim2=devv2./(pi/4*(diametre)^3);
v_adim2d=v2d./(pi/4*(diametre)^3);
devv_adim2d=devv2d./(pi/4*(diametre)^3);
v_adimE2=vE2./(pi/4*(diametre)^3);

figure (2)
h(1)=errorbar(Vb1(2:6),v_adim1(2:6),devv_adim1,'o','
    MarkerEdgeColor','k','Color','k','LineWidth',0.75);
hold on;
h(2)=errorbar(Vb1d(2:6),v_adim1d(2:6),devv_adim1d,'o','
    MarkerEdgeColor','k','MarkerFaceColor','k','Color','k','
    LineWidth',0.75);
hold on;
h(3)=plot(VbE1(2:6),v_adimE1(2:6),'*','MarkerEdgeColor','k',
    'Color','k');
hold on;
fplot(funct2,[0,2.5],'k')
hold on;
h(4)=errorbar(Vb2(2:6),v_adim2(2:6),devv_adim2,'s','
    MarkerEdgeColor','k','Color','k','LineWidth',0.75);
hold on;
h(5)=errorbar(Vb2d(2:6),v_adim2d(2:6),devv_adim2d,'s','
    MarkerEdgeColor','k','MarkerFaceColor','k','Color','k','
    LineWidth',0.75);
hold on;
h(6)=plot(VbE2(2:6),v_adimE2(2:6),'+','MarkerEdgeColor','k',
    'Color','k');
xlim([0 2.5])
xlabel('U_{SG}/f*\phi_{C}')
ylabel('$\bar{V}_{B}$','Interpreter','Latex','fontsize',14)
legend(h,'SimT 0.318 m/s','SimD 0.318 m/s','Exp 0.318 m/s','
    SimT 0.531 m/s','SimD 0.531 m/s','Exp 0.531 m/s','
    Location','southeast')
legend('boxoff')

%% Plot velocity
Um1=Usl1+Usg1; Um1(1)=0; %Mixture
Um2=Usl2+Usg2; Um2(1)=0;
Um=[Um1,Um2(2:6)];

uSimu=[u1,u2(2:6)];
```

```
200 uSimud=[u1d,u2d(2:6)];
201 uExp=[uE1,uE2(2:6)];
202 velocity_eq = fittype('Co*Um','coefficients',{'Co'},'
    dependent',{'uSimu'},'independent',{'Um'});
203 velocity_eqd = fittype('Co*Um','coefficients',{'Co'},'
    dependent',{'uSimud'},'independent',{'Um'});
204 velocity_eqE = fittype('Co*Um','coefficients',{'Co'},'
    dependent',{'uExp'},'independent',{'Um'});
205 u_eq1=fit(Um',uSimu',velocity_eq)
206 u_eq1d=fit(Um',uSimud',velocity_eqd)
207 uE_eq1=fit(Um',uExp',velocity_eqE);
208
209 figure(3)
210 axis([0 1.2 0 1.6])
211 plot(uE_eq1,'-k')
212 hold on;
213 plot(u_eq1,'--k')
214 hold on;
215 plot(u_eq1d,':k')
216 hold on;
217 h(1)=errorbar(Um1(2:6),u1(2:6),devu1,'o','MarkerEdgeColor','
    k','Color','k','LineWidth',0.75);
218 hold on;
219 h(2)=errorbar(Um1(2:6),u1d(2:6),devu1d,'o','MarkerEdgeColor'
    ,'k','MarkerFaceColor','k','Color','k','LineWidth',0.75);
220 hold on;
221 h(3)=plot(Um1(2:6),uE1(2:6),'*','MarkerEdgeColor','k','Color
    ','k');
222 hold on;
223 h(4)=errorbar(Um2(2:6),u2(2:6),devu2,'s','MarkerEdgeColor','
    k','Color','k','LineWidth',0.75);
224 hold on;
225 h(5)=errorbar(Um2(2:6),u2d(2:6),devu2d,'s','MarkerEdgeColor'
    ,'k','MarkerFaceColor','k','Color','k','LineWidth',0.75);
226 hold on;
227 h(6)=plot(Um2(2:6),uE2(2:6),'+','MarkerEdgeColor','k','Color
    ','k');
228 xlabel('U_{M} [m/s]')
229 ylabel('U_{B} [m/s]')
230 legend(h,'SimT 0.318 m/s','SimD 0.318 m/s','Exp 0.318 m/s','
    SimT 0.531 m/s','SimD 0.531 m/s','Exp 0.531 m/s','
    Location','southeast')
231 legend('boxoff')
232
233 %% Plot longitude
234 l_adim1=l1/diametre;
235 l_adim1d=l1d/diametre;
236 lE_adim1=lE1/diametre;
```

```
237
238 devl_adim1=devl1/diametre;
239 devl_adim1d=devl1d/diametre;
240
241 l_adim2=l2/diametre;
242 l_adim2d=l2d/diametre;
243 lE_adim2=lE2/diametre;
244
245 devl_adim2=devl2/diametre;
246 devl_adim2d=devl2d/diametre;
247
248 Vb=[Vb1,Vb2(2:6)];
249 Vbd=[Vb1d,Vb2d(2:6)];
250 VbE=[VbE1,VbE2(2:6)];
251 l_adim=[l_adim1,l_adim2(2:6)];
252 l_adimd=[l_adim1d,l_adim2d(2:6)];
253 lE_adim=[lE_adim1,lE_adim2(2:6)];
254 long_eq = fittype('C1+C2*Vb','coefficients',{'C1','C2'},'
    dependent',{'l_adim'},'independent',{'Vb'});
255 long_eqd = fittype('C1+C2*Vbd','coefficients',{'C1','C2'},'
    dependent',{'l_adimd'},'independent',{'Vbd'});
256 long_eqE = fittype('C1+C2*VbE','coefficients',{'C1','C2'},'
    dependent',{'lE_adim'},'independent',{'VbE'});
257 l_eq=fit(Vb','l_adim','long_eq)
258 l_eqd=fit(Vbd','l_adimd','long_eqd)
259 lE_eq=fit(VbE','lE_adim','long_eqE);
260
261 figure(4)
262 xlim([0 3])
263 ylim([0 4])
264 plot(l_eq,'--k')
265 hold on;
266 plot(l_eqd,':k')
267 hold on;
268 plot(lE_eq,'-k')
269 hold on;
270 h(1)=errorbar(Vb1(2:6),l_adim1(2:6),devl_adim1,'o','
    MarkerEdgeColor','k','Color','k','LineWidth',0.75);
271 hold on;
272 h(2)=errorbar(Vb1d(2:6),l_adim1d(2:6),devl_adim1d,'o','
    MarkerEdgeColor','k','MarkerFaceColor','k','Color','k','
    LineWidth',0.75);
273 hold on;
274 h(3)=plot(VbE1(2:6),lE_adim1(2:6),'*','MarkerEdgeColor','k',
    'Color','k');
275 hold on;
276 h(4)=errorbar(Vb2(2:6),l_adim2(2:6),devl_adim2,'s','
    MarkerEdgeColor','k','Color','k','LineWidth',0.75);
```

```
277 hold on;
278 h(5)=errorbar(Vb2d(2:6),l_adim2d(2:6),devl_adim2d,'s','
    MarkerEdgeColor','k','MarkerFaceColor','k','Color','k','
    LineWidth',0.75);
279 hold on;
280 h(6)=plot(VbE2(2:6),lE_adim2(2:6),'+','MarkerEdgeColor','k',
    'Color','k');
281 xlabel('U_{SG}/ f *\phi_{C}')
282 ylabel('$\bar{L}_{B}$','Interpreter','Latex','fontsize',14)
283 legend(h,'SimT 0.318 m/s','SimD 0.318 m/s','Exp 0.318 m/s','
    SimT 0.531 m/s','SimD 0.531 m/s','Exp 0.531 m/s','
    Location','southeast')
284 legend('boxoff')
```